RSA Blind Signatures with Public Metadata

Ghous Amjad
Google
gamjad@google.com

Kevin Yeo
Google
kwlyeo@google.com

Moti Yung
Google
moti@google.com

Abstract

Anonymous tokens are, essentially, digital signature schemes that enable issuers to provide users with signatures without learning the user inputs or the final signatures. These primitives allow applications to propagate trust while simultaneously protecting the user identity. They have become a core component for improving the privacy of several real-world applications including ad measurements, authorization protocols, spam detection, and VPNs.

In certain applications, it is natural to associate signatures with specific public metadata, ensuring that trust is only propagated with respect to only a certain set of users and scenarios. To solve this, we study the notion of anonymous tokens with public metadata. We present a variant of RSA blind signatures with public metadata where issuers may only generate signatures that verify for a certain choice of public metadata that is a modification of a scheme by Abe and Fujisaki [3]. Our protocol exclusively uses standard cryptography with widely available implementations. We prove security from the one-more RSA assumptions with multiple exponents that we introduce. Furthermore, we provide evidence that the concrete security bounds should be nearly identical to standard RSA blind signatures. We show that our protocol incurs minimal overhead over standard RSA blind signatures and report anonymous telemetry for a real-world deployment to showcase its scalability. Following our work, our protocol has been proposed as a technical specification in an IRTF internet draft [6].

Keywords

Anonymous Tokens, RSA Blind Signatures, Public Metadata

1 Introduction

Anonymous tokens are a powerful primitive that have been studied for decades under various names including anonymous credentials and blind signatures. They were first introduced by Chaum [19, 20] in the context of untraceable electronic cash. At a high level, the idea was that any bank/treasury would be able to attest the validity of money with the guarantee that no one would ever be able to trace the usage of the money even if the bank knows the identity of the user that was originally granted the money. This is just one example that exhibits the usefulness of anonymous tokens. In recent years, anonymous tokens have become a core component of many real-world applications such as private click measurement [61], privacy-preserving telemetry [37], fraud detection [30], avoiding repeated CAPTCHA solving [27] and modern VPNs [8, 44].

Proceedings on Privacy Enhancing Technologies 2025(1), 37–57

https://doi.org/10.56553/popets-2025-0004

There are typically three different roles for participants in anonymous token schemes. The user (or signature recipient or receiver) requests and receives the signature. The signer (or issuer) receives a blind signing request and issues a response that enables the user to recover the final signature. Finally, the verifier checks whether a signature is one that was correctly signed by the signer. Anonymous token schemes may be split into two types: designated or public verifiability. In the designated setting, verification requires the usage of the private key and thus certain parties with access to the private key must be explicitly assigned the role of verifier. In the public verifiability case, anyone with the public key can perform verification. In our work, we focus only on public verification. We choose to focus on public verification due to the flexibility that it offers when used in protocols where any party can verify signatures without needing the secret key. In particular, parties may act as verifiers without the ability to also issue signatures.

Anonymous tokens are required to satisfy three important properties: correctness, unforgeability and unlinkability. For correctness, it must be that a signature will be successfully accepted by a verifier if all parties were acting honestly by following the protocol correctly. Unforgeability describes the property that an adversary should not be able to create valid signatures without interacting with the signer e.g., an adversarial user performing ℓ blind signing protocols with a signer should not be able to create $\ell + 1$ distinct pairs of input message and valid signature. Both of these properties are required from any digital signature scheme. The last property of unlinkability formally describes the notion of anonymity with respect to an untrusted signer. In particular, the signer should be unable to link any blind signing request with any final signature. Suppose that the signer answers ℓ blind signing requests. Afterwards, the signer receives the ℓ final signatures that are randomly permuted. Then, the signer is unable to link any request with a signature with probability better than $1/\ell$ even if the signer may have maliciously chosen the system's parameters.

Anonymous Tokens with Public Metadata. In this work, we study anonymous tokens where each signature will be tied to a some public metadata (also known as partially blind signatures [2–4]). Public metadata is a powerful tool that enables additional information that may be integral during verification. In standard anonymous tokens (without public metadata), the verifier only receives the signature and plaintext message. Furthermore, the signer issuing the signature never sees the plaintext message to maintain anonymity. In other words, there is no way for a signer to properly check and embed metadata into a signature that can be later used by the verifier. This feature ends up being quite useful in multiple applications as we will show later. Anonymous tokens with public metadata allow the user and signer to jointly agree on metadata explicitly encoded into each token. This metadata will be public to all parties (the input message remains hidden from the signer for anonymity). Additionally, the verifier will check signatures for a

specific choice of public metadata. The primitives must guarantee that a signature will only be accepted if the verifier uses the correct public metadata used in signing.

Public Metadata for Privacy-Enhancing Technologies. The ability to augment anonymous tokens with public metadata serves as a way to propagate trust for a specific subset of settings that is useful for designing privacy-enhancing technologies (PETs). In most applications using anonymous tokens, the signer is typically responsible for checking that the user is permitted to receive tokens. During verification, the verifier no longer has any information about the user and, thus, may only check that tokens are valid. Public metadata enables propagating this permission in a fine-grained manner. The verifier can check this public metadata with trust that the signer embedded the public metadata. We present two real-world example applications where public metadata is critical.

Application 1: Geolocality-based VPN. As a real-world application, we can consider VPN where each user purchases access for a specific geolocality such as a country. Whenever a user accesses the VPN, it is critical that the user's anonymity is maintained. However, the VPN provider also needs to check that the accessing user has paid for access in the specified country. In fact, this is the exact problem encountered by GoogleOne VPN [44]. We show that this can be solved with public metadata for anonymous tokens. To access the VPN, a user is first required to authenticate to prove that they have purchased access for a specific country. Afterwards, the user will receive several anonymous tokens where the public metadata is the country for authorized access. When the user accesses the VPN, the user will redeem anonymous tokens. The VPN provider can verify that each token was created for the correct country before permitting VPN access.

Application 2: Short-Lived Anonymous Tokens. As another example, we consider the general setting where anonymous tokens should have short expiration times or may be redeemed within a short time period (for example, 1 hour). This could be useful for many applications such as timed event access or promotions with expiration. A trivial approach is to utilize different keys for each expiration time or valid time period (such as one key per hour). However, managing a large number of keys that is a very challenging problem in real-world deployments [17, 28, 50]. For example, it requires a large number of key rotations such as once every hour. Typically, key rotations are performed after a certain number of operations to avoid security degradation or mitigating accidental key exposure. In this case, we are abusing key rotation to enable expirations. The large number of key rotations may introduce additional problems (such as missing keys, exposure, etc). It is advantageous to keep the key set small to avoid these potential issues. Instead, we can use public metadata to avoid unnecessary key rotations. When issuing a token, the signer finds the correct expiration time (or valid time period) for each user and embeds it as public metadata. Then, the verifier will only permit access to the user if the anonymous token is redeemed at the right time period.

1.1 Our Contributions

RSA Blind Signatures with Public Metadata. As our main contribution, we present a blind signature with public metadata based on the RSA assumption that is a modification of a scheme introduced by Abe and Fujisaki [3]. The construction uses only standard cryptography widely available across most platforms while avoiding more complex and less supported operations (such as pairings). We formally prove unforgeability from a variant of the one-more RSA assumption as well as unconditional unlinkability. Additionally, we derive concrete security bounds to enable picking parameters in real-world deployments. We note that the prior work [3] did not have formal definitions or security proofs. One of this paper's main contributions is proving that an adjusted version of their scheme is secure. Finally, we show that our protocol incurs only slight overhead over standard RSA blind signatures. We note that there are prior schemes for public metadata that rely on cryptography with more limited production availability such as pairings [56]. In Appendix I, we compare the availability of the necessary RSA operations for our scheme to pairings in production cryptography libraries. **IRTF Draft.** Following our paper, the protocols in this work have been proposed in a CFRG specification in the IRTF [6] due to the interest of practitioners for real-world applications. By constructing practical anonymous tokens with public metadata using widely available production cryptography, our construction has enabled new applications for PET designers. For example, there are already work streams to build real-world architectures relying on our construction in the IETF Privacy Pass working group (such as [35]). **Experimental Evaluation and Deployment Telemetry.** We perform experimental evaluation to show our protocol incurs minimal overhead over standard RSA blind signatures. Additionally, our protocol has been deployed in an application for GoogleOne VPN. We report on anonymous real-world telemetry to showcase the scalability of our protocol to millions of users.

1.2 Technical Overview

Before we present our protocol for RSA blind signatures with public metadata, we first show some failed approaches which try to enable public metadata in an anonymous tokens scheme and outline their downsides. This provides insights into our design choices.

Naive Solution: Multiple Keys. The first idea is to generate a new key for each option of public metadata D. The main downside of this approach is that key management becomes significantly more challenging with large key sets. As mentioned earlier, key management is one of the most difficult aspects in real-world deployments [17, 28, 50]. Instead, we want a protocol where the number of key pairs does not grow with the public metadata universe.

Failed Solution: Embedding into Message. Another approach is to append the public metadata D to the message M to obtain message $M' = M \,||\, D$. Afterwards, the anonymous token scheme remains identical using new message M'. Unfortunately, this approach does not work due to the anonymity guarantees. In particular, the metadata D must be known to the signer during signing. For anonymity, the message $M' = M \,||\, D$ must be hidden from the signer. Therefore, the signer cannot verify that they are signing with respect to a specific public metadata D and, hence, this approach fails to satisfy the necessary unforgeability requirements.

Existing Solutions. There are other potential solutions to support public metadata in anonymous tokens. Silde and Strand [56] present a pairing-based protocol based on BLS signatures [15]. In general, pairings are computationally expensive and not widely available in

production libraries. Several works [38, 57] consider pairing-free solutions, but require three moves (resulting in multiple roundtrips during blind signing). There are other options such as Albrecht *et al.* [5] using torus-based fully homomorphic encryption (FHE), which is even more expensive than pairings. We consider two-move protocols using efficient cryptography to enable easier adoption.

Our Protocol. Using the above approaches as guidance, we can determine requirements necessary for our construction. First, we should only use a single key independent of the public metadata universe. Additionally, we still need the public metadata to be viewable by the signer during blind signing. Finally, we should avoid any complex cryptography that is not widely supported.

We first revisit the original RSA blind signature protocol. Recall that standard RSA blind signatures utilize a modulus that is the product of two primes $N = pq$ along with a public exponent e and a private exponent d such that $d = e^{-1} \mod \phi(N)$. The signer's private key consists of d while the public key is (N, e). All operations are done in $\mathbb{Z}_N^*$. To perform blind signing for a message M, the user computes $A = H_{\mathcal{M}}(M) \cdot R^e$ where $H_{\mathcal{M}}(M)$ hashes the message M to an element of $\mathbb{Z}_N^*$ and R is a uniformly random element of $\mathbb{Z}_N^*$. The signer computes $B = A^d = H_{\mathcal{M}}(M)^d \cdot R^{ed} = H_{\mathcal{M}}(M)^d \cdot R$. Finally, the user computes final signature $S = B \cdot R^{-1} = H_{\mathcal{M}}(M)^d \cdot R \cdot R^{-1} = H_{\mathcal{M}}(M)^d$. To verify S for message M using (N, e), one simply computes S^e and checks if it equals to $H_{\mathcal{M}}(M)$.[1]

We take insight from the first failed approach that used different keys for each metadata along with prior work of Abe and Fujisaki [3]. Rather than using different keys, we use a hash function H_{MD} to enable generating keys for each metadata D. In more detail, we will set $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$ as the public exponent as done in the scheme presented in [3]. In our scheme, we use a random string as a salt in H_{MD} but omit it here for simplicity. Blind signing for the user remains similar using e_{MD} as opposed to e and putting D into the message hash to obtain $A = H_{\mathcal{M}}(M \mid\mid D) \cdot R^{e_{\mathsf{MD}}}$. Now, the signer first computes $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$. We augment the private key to contain $\phi(N)$ allowing the signer to compute the inverse $d_{\mathsf{MD}} = (e_{\mathsf{MD}})^{-1} \mod \phi(N)$. Afterwards, the signer uses d_{MD} as the private exponent to return $B = H_{\mathcal{M}}(M \mid\mid D)^{d_{\mathsf{MD}}} \cdot R$ to the user. Finally, the user removes R to obtain $S = H_{\mathcal{M}}(M \mid\mid D)^{d_{\mathsf{MD}}}$.

Unfortunately, the above construction has a slight problem. In particular, we assumed that the output $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$ is always invertible modulo $\phi(N)$ and, thus, co-prime to $\phi(N)$. For standard RSA modulus $N = pq$ where p and q are distinct primes, it is not necessarily the case that a random element would be invertible modulo $\phi(N) = (p-1) \cdot (q-1)$. To solve this problem, we can use strong RSA moduli where we require that $N = pq$ such that both p and q are safe primes meaning that $p = 2p' + 1$ and $q = 2q' + 1$ such that both p' and q' are also prime numbers. If N is a strong RSA modulus, then we know that $\phi(N) = (p-1) \cdot (q-1) = 4p'q'$. Therefore, an element $x \in \mathbb{Z}_{\phi(N)}$ has an inverse as long as x is odd and not divisible by p' and q'. If we assume that both p and q are κ-bit prime numbers, then we can guarantee that the output of H_{MD} is always invertible in $\mathbb{Z}_{\phi(N)}$ using the following modifications. First, we ensure that H_{MD} always outputs an odd number. Next, we make sure that H_{MD} always outputs elements of length at most $\kappa - 2$ bits. As p and q are κ bits, we know that both p' and q' are at least $\kappa - 1$ bits. As a result, we can guarantee that H_{MD} is always invertible modulo $\phi(N)$ as it is odd and always smaller than both p' and q'. Even though similar ideas were previously presented in [3], no formal security proofs existed prior to our work.

One-More RSA Inversion with Multiple Exponents. Before proving the security of our protocol, we first define an extension of the "one-more" RSA assumption [11]. We explore several natural ways to define RSA assumptions with respect to multiple exponents before arriving at the weakest form that may be used to prove security of our protocol. To our knowledge, this is the first exploration of one-more RSA inversion assumptions with multiple exponents. Finally, we also explore connections with the strong RSA assumption where the adversary picks the public exponent.

Security of Our Protocol. Finally, we prove the security of our new protocol with public metadata. We start with proving concrete security of the non-blind variant from the RSA assumption. In particular, we show that our new protocol has similar concrete security guarantees for unforgeability as standard RSA signatures. We note that our subtle modification where the underlying hash $H_{\mathcal{M}}(M \mid\mid D)$ includes both M and D is integral in our security proof. Next, we prove the security of our main RSA blind signatures with public metadata protocol. Using the one-more multi-exponent RSA inversion assumptions, we are able to prove the unforgeability of our protocol. For anonymity, we adapt the techniques introduced by [40] to prove that our protocols satisfy unlinkability even in the presence of maliciously generated keys.

Underlying Cryptography Operations. One benefit of starting from RSA blind signatures is that all the underlying cryptographic operations are widely supported. The only additional functionalities required by our protocol is the ability to hash strings to random numbers for H_{MD} and generating random safe primes (only needed for key generation). These are cryptographic operations that are widely supported for production usage. In contrast, pairings (relied upon by prior works such as [56]) are not yet available in most widely used production cryptography libraries. Although, recent interest in pairing-based cryptography (such as IRTF draft [14]) may lead to pairings being more widely available in the future. See Appendix I for availability of operations in production libraries.

1.3 Related Work

BLS with Public Metadata. Silde and Strand [56] also studied anonymous tokens with public metadata. For the setting of public verifiability, the authors presented a construction from pairings based on the BLS signature scheme [15] along with ideas from [27, 63]. Comparing with [56], our protocol uses RSA-based cryptography as opposed to pairing-based cryptography. RSA-based cryptography is more readily available on all platforms compared to pairing-based cryptography enabling easier adoption. We note that recent IRTF work [14] may lead to pairings being more widely available for production usage in the future. See Appendix I for comparison of production availability of pairings compared to the RSA operations for our scheme.

Partial OPRF with Public Metadata. Several works have studied partially oblivious pseudo-random functions (OPRF) including [56, 60, 63] that essentially allows public metadata for OPRF evaluations.

[1] We assume here that $H_{\mathcal{M}}$ is deterministic. However, there are randomized message encodings with more complex verification. See Section 5.1 for more details.

We note that there was a proof flaw in [63] that was fixed in [60] and all schemes are secure.

RSA Blind Signatures with Strong Moduli. We note that similar variants of RSA blind signatures with public metadata has appeared in the past [2, 3]. However, the security was proven heuristically. One can view the main contribution of our work as proving a modified variant of the scheme in [3] to be secure along with concrete security bounds. Furthermore, we combine recent work [40] to obtain protection against maliciously generated keys. Other works also used strong RSA modulus such as threshold signatures [54] and integer commitments [25].

Anonymous Tokens with Private Metadata. Recent work [39] studied anonymous tokens augmented with a single private metadata bit along with publicly verifiable variants [13, 46]. The metadata bit is explicitly specified only by the signer and observable by the verifier while hidden from the user. Another recent work [18] studied augmenting private metadata into anonymous tokens by using algebraic MACs. To our knowledge, blind signatures cannot be used to instantiate anonymous tokens with private metadata. We leave it as future work to support private metadata bits using RSA-style signatures.

2 Anonymous Tokens

We present the formal definitions for anonymous token schemes which will also work for non-anonymous protocols with small modifications. We will solely focus on the settings of public metadata and public verifiability. Our definitions are interchangeable with the notion of partially blind signature schemes [2, 4] and we explicitly focus on two-move (message-response structure) blind signatures. Non-two-move signatures exist such as [4] but due to the the extra communication round-trip we do not focus on them in this work.

Definition 2.1. A token scheme with public metadata Tok that is publicly verifiable is a tuple of efficient algorithms Tok = (Setup, Blind, Sign, Finalize, Verify).

(1) $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Tok.Setup}(1^\lambda)$: The setup algorithm receives the security parameter λ as input and outputs a pair of public and private keys $(\mathsf{pk}, \mathsf{sk})$.
(2) $(\mathsf{st}, B_M) \leftarrow \mathsf{Tok.Blind}(M, D, \mathsf{pk})$: The blinding protocol is run by the user who receives the plaintext message M, public metadata D and public key pk. The output B_M is a blinded version of the message M under public metadata D and some state st.
(3) $S' \leftarrow \mathsf{Tok.Sign}(B_M, D, \mathsf{pk}, \mathsf{sk})$: The signing protocol is run by the signer who receives the blinded message B_M, public metadata D, public and private keys $(\mathsf{pk}, \mathsf{sk})$. The output S' is a signature on B_M under public metadata D which needs to be finalized by the user.
(4) $S \leftarrow \mathsf{Tok.Finalize}(\mathsf{st}, S', M, D, \mathsf{pk})$: The user runs the Finalize protocol on the user state st, the signer's response S', plaintext message M, public metadata D and public key pk. The output S is a signature of the message M under public metadata D.
(5) $b \leftarrow \mathsf{Tok.Verify}(S, M, D, \mathsf{pk})$: The verification algorithm receives the signature S, the plaintext message M, public metadata D and public key pk and outputs a bit $b \in \{0, 1\}$.

Tok satisfies the correctness properties if, for all choices of messages M and public metadata D, the following probability

$$\Pr\left[\mathsf{Verify}(S, M, D, \mathsf{pk}) \neq 1 \;:\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{st}, B_M) \leftarrow \mathsf{Blind}(M, D, \mathsf{pk}) \\ S' \leftarrow \mathsf{Sign}(B_M, D, \mathsf{pk}, \mathsf{sk}) \\ S \leftarrow \mathsf{Finalize}(\mathsf{st}, S', M, D, \mathsf{pk})\end{array}\right]$$

is at most negligible in λ, $\mathsf{negl}(\lambda)$.

The Blind, Sign, Finalize protocols can be viewed as a single round-trip protocol between the user and signer. One could generalize the above definition to encompass multiple round, interactive protocols. However, the structure of the Blind, Sign, Finalize protocol will be useful for proving unforgeability of our schemes. Therefore, we only focus on this structure throughout our work.

To obtain the standard definition of non-anonymous tokens, we can simply restrict the functionality of Blind and Finalize. Blind will return $(\mathsf{st} \leftarrow \perp, B_M \leftarrow M)$ and Finalize will return $S \leftarrow S'$.

Relation with Blind Signatures. In our work, we treat anonymous tokens synonymously with blind signatures. In general, blind signatures may be used to instantiate anonymous tokens (as we do in our work). However, anonymous tokens may be more general and may allow for designated (secret key) verifiers as studied in [35]. As we focus exclusively on public verification with public metadata, Definition 2.1 is equivalent to partially blind signatures [2, 4]. We also note that it is unknown whether standard blind signatures may be used to instantiate private metadata [39].

2.1 Unforgeability

Unforgeability, the first cryptographic guarantee provided by anonymous tokens, states that no party is able to generate signatures that will correctly verify unless they have access to the private key. For our work, we will consider a modification of the standard definition of unforgeability for blind signatures from [52]. This definition is known as *strong one-more unforgeability* as it was a strengthening of previous definitions in [11, 48]. We modify this definition of unforgeability to enable public metadata in the following way. In prior definitions without public metadata, the adversary wins if it can construct $\ell + 1$ signatures using at most ℓ signing oracle queries. In our game, the adversary succeeds if it is able to construct at least $\mathsf{cnt}_D + 1$ signatures for any choice of public metadata D using at most cnt_D signing oracle queries with D. This even covers the case when a signature is forged for some D without ever sending a signing query with D. The security game can be found in Figure 1 and we present the formal definition below.

Definition 2.2 ((ϵ, t, ℓ)-Strong One-More Unforgeability). Let λ be the security parameter and consider the game $\mathrm{G}^{\mathsf{SOMUF}}_{\mathsf{Tok},\mathcal{A}}$ in Figure 1. A token scheme Tok is (ϵ, t, ℓ)-strong one-more unforgeable if, for any adversary $\mathcal{A}$ that runs in probabilistic time t and makes at most ℓ signing queries, then $\Pr[\mathrm{G}^{\mathsf{SOMUF}}_{\mathsf{Tok},\mathcal{A}}(\lambda) = 1] \leq \epsilon$.

Difference with Previous Definition. In a slightly different definition in [56], the adversary is able to make at most ℓ signing queries for any metadata. Then, the adversary wins the game if they fix some metadata D and can create $\ell + 1$ valid signatures. We instead bound the total number of signing queries by ℓ and the adversary wins the game if they are able to generate $\mathsf{cnt}_D + 1$ valid signatures

Game $\mathrm{G}^{\mathsf{SOMUF}}_{\mathsf{Tok},\mathcal{A}}(\lambda)$:	Oracle $O^{\mathsf{Sign}}(M, D)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Tok.Setup}(1^\lambda)$ Initialize $\mathsf{cnt}_D \leftarrow 0$ for all choices of public metadata D. $D, (S_i, M_i)_{i\in[x]} \leftarrow \mathcal{A}^{O^{\mathsf{Sign}}}(\mathsf{pk})$ **Return** 1 if and only if all the following hold: - $\mathsf{cnt}_D < x$ - $\forall i \neq j \in [x], S_i \neq S_j \vee M_i \neq M_j$ - $\forall i \in [x], \mathsf{Tok.Verify}(S_i, M_i, D, \mathsf{pk}) = 1$	$\mathsf{cnt}_D \leftarrow \mathsf{cnt}_D + 1$ **Return** $\mathsf{Tok.Sign}(M, D, \mathsf{pk}, \mathsf{sk})$.

Figure 1: Strong One-More Unforgeability (SOMUF) Game with Public Metadata.

for any metadata D while making at most cnt_D oracle queries for metadata D. We do this to derive concrete security bounds with respect to ℓ as opposed to [56], where the adversary may continue to make ℓ oracle queries for new choices of public metadata D. Furthermore, we note that our definition is no weaker than the prior definition. If any adversary can produce $\ell + 1$ signatures using at most ℓ oracle queries, then there must exist some public metadata D such that the adversary has more valid signatures than oracle signing queries (see Appendix B for more details).

Multiple Signers. Our definition only considers a single signer whereas general settings may consider multiple signers. It turns out that the single and multiple signers are equivalent up to some factors that depend only on the number of signers. Furthermore, the strong one-more unforgeability is identical in either the single or multiple signer settings for two-move protocols (as studied in this paper). So, it is sufficient to focus security proofs with respect to a single signer. For more details, we point readers to [40].

Anonymous vs. Non-Anonymous Tokens. We will use the same definition for both token types and slightly overload notation for convenience. For non-anonymous tokens, the input to the signing oracle will be the plaintext message. For anonymous tokens, the input will be the blinded (i.e., encrypted) message.

2.2 Unlinkability

Unlinkability or anonymity, the second cryptographic guarantee provided by anonymous tokens, states that it must be impossible to determine the blind signing request that was utilized to create a signature even by the signer that views the signing interactions, the final signature as well as the corresponding input message. We modify the definition of unlinkability from [1] for single-round schemes to encompass public metadata. In this game-based definition, the adversary picks the public key, two messages M_0, M_1 as well as its choice of public metadata D. The challenger will first blind both messages and randomly permute their order. The blinded messages are submitted to the adversary in this order, who will return signatures on them. Finally, the challenger finalizes the adversary's responses to obtain the final signatures. Those are given to the adversary in the original message order along with input messages. To win the game, the adversary has to guess the correct permutation of the blinded messages.

Definition 2.3 ((ϵ, t)-Unlinkability). Let λ be the security parameter and consider game $\mathrm{G}^{\mathsf{UNLINK}}_{\mathsf{Tok},\mathcal{A},q}$ in Figure 2. Token scheme Tok satisfies (ϵ, t)-unlinkability if, for any adversary $\mathcal{A}$ running in time t, then: $|\Pr[\mathrm{G}^{\mathsf{UNLINK}}_{\mathsf{Tok},\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathrm{G}^{\mathsf{UNLINK}}_{\mathsf{Tok},\mathcal{A}}(\lambda, 1) = 1]| \leq \epsilon$.

Discussion about Public Metadata. In our definition, the adversary is able to choose any public metadata. However, the same public metadata must be used for signing both messages. This is necessary as if the adversary may pick different public metadata for each message then it can trivially win the game. In practice, this implies that anonymity only applies to all groups of messages signed with the same public metadata.

Game $\mathrm{G}^{\mathsf{UNLINK}}_{\mathsf{Tok},\mathcal{A}}(\lambda, b)$:

Adversary $\mathcal{A}$ outputs $(\mathsf{pk}, M_0, M_1, D) \leftarrow \mathcal{A}(1^\lambda)$.
For $i \in \{0, 1\}$:
  Challenger C computes $(B_i, \mathsf{st}_i) \leftarrow \mathsf{Tok.Blind}(\mathsf{pk}, M_i, D)$.
C samples uniformly random bit $b \leftarrow_R \{0, 1\}$.
$\mathcal{A}$ receives B_b, B_{1-b} and computes S'_b, S'_{1-b}.
For $i \in \{0, 1\}$:
  C computes $S_i \leftarrow \mathsf{Tok.Finalize}(\mathsf{pk}, \mathsf{st}_i, S'_i, M_i, D)$.
If $\mathsf{Tok.Verify}(M_0, S_0, D, \mathsf{pk}) = 1 \wedge \mathsf{Tok.Verify}(M_1, S_1, D, \mathsf{pk}) = 1$:
  C sends (M_0, S_0), (M_1, S_1) to $\mathcal{A}$.
Else:
  C sends $\perp$ to $\mathcal{A}$.
$\mathcal{A}$ outputs a bit b'.

Figure 2: Unlinkability (UNLINK) Game.

Adversarial Signer. Our choice of unlinkability definitions requires anonymity even against malicious adversarial signers. In particular, we note that the adversary is able to choose any public key and is free to compute signatures arbitrarily.

Necessity of Successful Verification. In our definition, we note that the adversary is only permitted to see the resulting signatures if both signatures successfully verified. This is necessary to rule out one naive strategy for the adversary where it would issue a valid signature in response to one of the queries but not the other.

3 Prior RSA Assumptions

In this section, we outline the standard RSA assumption and one-more RSA inversion assumptions that are relevant to our work.

Strong RSA Modulus. In our work, we will focus on a special subset of RSA moduli that are strong RSA modulus. All strong RSA modulus $N = pq$ are the product of two safe primes that enables structure to the multiplicative group modulo $\phi(N)$.

Definition 3.1 (Strong RSA Modulus). An integer N is a strong RSA modulus $N = p \cdot q$ where each of p and q are distinct safe primes. In other words, $p = 2p' + 1$ and $q = 2q' + 1$ where all of p, p', q, q' are distinct prime numbers. Therefore, $\phi(N) = 4p'q'$.

We assume that both p' and q' are $\kappa - 1$ bits (p and q are κ bits). Since $\phi(N) = 4p'q'$, we will use that any random odd number from the set $[3, 2^{\kappa-2}]$ will be co-prime to $\phi(N)$.

RSA Assumption. We will denote generating a random modulus, the public exponent distribution and secret parameters by $(N, \mathcal{D}_N, p, q) \leftarrow_R \mathsf{Gen}(1^\lambda)$. The prime bit-length κ is chosen to

Game $\mathrm{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$:
$(N, \mathcal{D}_N, (p,q)) \leftarrow_R \mathsf{Gen}(1^\lambda)$
$e \leftarrow_R \mathcal{D}_N, X \leftarrow_R \mathbb{Z}^*_N$
$Y \leftarrow \mathcal{A}(N, e, X)$
Return 1 if and only if $Y^e = X \bmod N$.

Figure 3: RSA Game.

obtain λ bits of security. At a high level, the RSA assumption assumes that any probabilistically polynomial time (PPT) adversary $\mathcal{A}$ given a RSA modulus N and public exponent e coprime to $\phi(N)$ is unable to compute the e-th root modulo N for a random element $X \leftarrow_R \mathbb{Z}^*_N$. In other words, the PPT adversary $\mathcal{A}$ is unable to compute $X^d \bmod N$ where $d = e^{-1} \bmod \phi(N)$. We will commonly refer to $X^e \bmod N$ as an RSA encryption and $X^d = X^{1/e} \bmod N$ as an RSA decryption throughout our work. In general, the RSA assumption is a class of assumptions that is parameterized by some distribution $\mathcal{D}_N$ determining how to pick the public exponent e. Thus, the Gen algorithm also outputs a distribution $\mathcal{D}_N$ to sample e. Furthermore, for concrete security bounds that may be used for guiding practical implementations, we consider a more fine-grained version of the RSA assumption that is additionally parameterized by the running time t and advantage ϵ of the adversary.

Definition 3.2 ((ϵ, t)-RSA Assumption). Let λ be the security parameter and consider the game $\mathrm{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 3. The (ϵ, t)-RSA assumption is true for Gen if for any PPT adversary $\mathcal{A}$ that runs in time t, then $\Pr[\mathrm{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon$.

In our work, we will focus on the setting when Gen always outputs strong RSA modulus N. We do not use the safe primes for reasons related to strengthening the RSA assumption. Instead, we use safe primes to utilize the additional structure in the multiplicative group $\mathbb{Z}^*_{\phi(N)}$. In particular, we show that safe primes enable an efficient and simple method of hashing public metadata to a large subset of elements in $\mathbb{Z}^*_{\phi(N)}$ without requiring knowledge of $\phi(N)$.

Definition 3.3 ((ϵ, t)-RSA Assumption for Strong Modulus). Let λ be the security parameter and consider the game $\mathrm{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 3. The (ϵ, t)-RSA assumption is true for Gen such that Gen only outputs strong RSA modulus and, if for any PPT adversary $\mathcal{A}$ that runs in time t, then $\Pr[\mathrm{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon$.

As the RSA assumption for strong modulus considers only a subset of possible modulus, it seems natural that it is no weaker than the standard RSA assumption (Theorem 3.4). We only assume that Gen generates N as the product of two uniformly random κ-bit primes, which is common practice (see BoringSSL [33] for example). See Appendix A for the proof of Theorem 3.4.

Theorem 3.4. *If the (ϵ, t)-RSA Assumption (Definition 3.2) is true and* Gen *generates modulus N as the product of two uniformly random κ-bit primes, then the $(O(\epsilon \cdot \kappa^2), t)$-RSA Assumption for Strong Modulus (Definition 3.3) is also true.*

In our proofs, we will prove the unforgeability of our RSA (non-blind) signatures with public metadata using the RSA assumption for strong modulus. Using Theorem 3.4, one can then re-interpret all our results as assuming the standard RSA assumption. Finally, we note that the above reduction lets us interchangeably use various RSA assumptions for arbitrary modulus as well as strong modulus at the costs of an $O(\epsilon\kappa^2)$ multiplicative factor.

In practical implementations, the distribution $\mathcal{D}_N$ is typically fixed for all modulus N. The most common public exponents used in RSA implementations are 3 and 65537. On the other hand, theoretical works will typically consider $\mathcal{D}_N$ to be uniform over some subset of $\mathbb{Z}^*_{\phi(N)}$ such as [23]. We will follow the same approach in our work and assume $\mathcal{D}_N$ outputs random odd γ-bit integers where γ is chosen later in our constructions.

One-More RSA Inversion Assumption. Security of RSA blind signature schemes have been proven by using of a class of computational problems known as one-more RSA inversion problems introduced in [11]. In these problems, the adversary is given access to a decryption oracle O^{RSA} and a challenge oracle O^X. The decryption oracle takes as input $X \in \mathbb{Z}^*_N$ and returns $O^{\mathsf{RSA}}(X) = Y$ where $Y^e = X \bmod N$. The challenge oracle O^X will return random elements from $\mathbb{Z}^*_N$ as random challenge targets. The adversary only succeeds if it inverts $\ell + 1$ challenge targets while making at most ℓ queries to the decryption oracle O^{RSA}. We define chosen-target RSA inversion game in Figure 4 along with the following definition that were both introduced in [11]. In this definition, *chosen-target* implies that the adversary may choose to invert any of the targets output by the oracle O^X.

Definition 3.5 ((ϵ, t, ℓ)-Chosen-Target RSA Inversion Assumption). Let λ be the security parameter and consider the game $\mathrm{G}^{\mathsf{CT\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 4. The (ϵ, t, ℓ)-chosen-target RSA inversion assumption is true for Gen if, for any adversary $\mathcal{A}$ that runs in time t and makes at most ℓ decryption queries, then $\Pr[\mathrm{G}^{\mathsf{CT\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon$.

4 RSA with Multiple Exponents

In this section, we introduce and explore various generalizations of the RSA assumption with multiple exponents. Afterwards, we further extend them to one-more variants (following the definitional extensions for the RSA assumption in [11]). To our knowledge, we are unaware of prior formal definitions of multi-exponent variants.

4.1 Multi-Exponent RSA Assumption

We start by extending the standard RSA assumption from one challenge exponent e to a set of ℓ exponents, $e_1, \ldots, e_\ell$. The adversary wins the game if it can decrypt a random target for any of the ℓ exponents provided in the challenge.

Definition 4.1 ((ϵ, t, ℓ)-Multi-Exponent RSA Assumption). Let λ be the security parameter and consider $\mathrm{G}^{\mathsf{ME\text{-}RSA}}_{\mathsf{Gen},\mathcal{A},\ell}(\lambda)$ in Figure 5. The (ϵ, t, ℓ)-multi-exponent RSA assumption is true for Gen, if for any adversary $\mathcal{A}$ that runs in time t, then $\Pr[\mathrm{G}^{\mathsf{ME\text{-}RSA}}_{\mathsf{Gen},\mathcal{A},\ell}(\lambda) = 1] \leq \epsilon$.

As this provides more flexibility for the adversary, it is a stronger assumption than the standard RSA assumption. However, we can show that the two assumptions differ by a multiplicative ℓ factor. The proof may be found in Appendix C.

Theorem 4.2. *If the (ϵ, t)-RSA assumption for strong modulus (Def. 3.3) is true with $\mathcal{D}_N$ choosing uniformly random exponents from $[3, e_{\max}]$ where $e_{\max} \leq 2^{\kappa-2}$, then the $(\epsilon \cdot \ell, t - O(\ell), \ell)$-multi-exponent RSA assumption (Def. 4.1) is with $\mathcal{D}_N$.*

Game $\mathrm{G}^{\text{CT-RSA}}_{\text{Gen},\mathcal{A}}(\lambda)$:	Oracle $O^{\text{RSA}}(X)$:	Oracle $O^{X}()$:
$(N, \mathcal{D}_N, (p,q)) \leftarrow_R \text{Gen}(1^\lambda)$ $e \leftarrow_R \mathcal{D}_N$ $\phi(N) \leftarrow (p-1)(q-1)$ $\text{cnt} \leftarrow 0, \mathcal{S}_X \leftarrow \emptyset$ $(X_i, Y_i)_{i\in[\text{cnt}+1]} \leftarrow \mathcal{A}^{O^{\text{RSA}}, O^{X}}(N, e)$ **Return** 1 if and only if all the following hold: - $\forall i \neq j \in [\text{cnt}+1], X_i \neq X_j$ - $\forall i \in [\text{cnt}+1], X_i \in \mathcal{S}_X$ - $\forall i \in [\text{cnt}+1], Y_i^e = X_i \bmod N$	$\text{cnt} \leftarrow \text{cnt} + 1$ $d \leftarrow e^{-1} \bmod \phi(N)$ $Y \leftarrow X^d \bmod N$ **Return** Y	$X \leftarrow_R \mathbb{Z}_N^*$ $\mathcal{S}_X \leftarrow \mathcal{S}_X \cup \{X\}$ **Return** X

Figure 4: Chosen-Target RSA Inversion Game.

Game $\mathrm{G}^{\text{ME-RSA}}_{\text{Gen},\mathcal{A},\ell}(\lambda)$:
$(N, \mathcal{D}_N, (p,q)) \leftarrow_R \text{Gen}(1^\lambda)$ $(e_1, \ldots, e_\ell) \leftarrow_R \mathcal{D}_N$ $X \leftarrow_R \mathbb{Z}_N^*$ $(i, Y) \leftarrow \mathcal{A}(N, e_1, \ldots, e_\ell, X)$ **Return** 1 if and only if $i \in [l]$ and $Y^{e_i} = X \bmod N$.

Figure 5: Multi-Exponent RSA Game. All differences with the RSA game are in blue.

Tighter Reductions. We show that one can obtain a tighter reduction with no security loss if we assume a slightly different distributions of exponents in the multi-exponent RSA assumption (that also appeared in [3]). See Appendix F for the different exponent distribution and the proof of the reduction.

4.2 One-More Multi-Exponent RSA Assumption

We start from the chosen-target variant of the one-more RSA assumption (Figure 4). We will modify this game to obtain a one-more variant of the multi-exponent RSA assumption. As the first step, we will also create an exponent oracle O^{exp} that will output challenge exponents. Next, we generalize the decryption oracle, O^{RSA}, to receive both an element X and an exponent e. However, O^{RSA} makes the restriction that the input exponent e must be a challenge exponent output by O^{exp}. This immediately leads to the first natural definition of a one-more multi-exponent RSA assumption where an adversary makes at most cnt_e decryption oracle queries for some exponent e and must output $\text{cnt}_e + 1$ valid decryptions of the form $(X_i, Y_i)_{i\in[\text{cnt}_e+1]}$ satisfying that $Y_i^e = X_i \bmod N$ where e must be some challenge exponent output by O^{exp}. We keep the same restriction as the one-more RSA assumption that the targets X_i must be valid outputs from the message oracle O^X. We formally define this game in Figure 6. In this definition, we use *restricted-exponent* to denote the fact that the decryption oracle O^{RSA} restricts the adversary to only pick exponents output by O^{exp}.

Definition 4.3 ($(\epsilon, t, \ell, \ell')$-Chosen-Target, Restricted-Exponent RSA Inversion Assumption). Let λ be the security parameter and consider the game $\mathrm{G}^{\text{CT-RE-RSA}}_{\text{Gen},\mathcal{A}}(\lambda)$ in Figure 6. The $(\epsilon, t, \ell, \ell')$-chosen-target, restricted-exponent RSA inversion assumption is true for Gen, if for any adversary $\mathcal{A}$ that runs in time t, makes ℓ decryption queries, and makes ℓ' exponent queries, $\Pr[\mathrm{G}^{\text{CT-RE-RSA}}_{\text{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon$.

Algebraically Restricted Exponents. We present a variant where the exponent oracle will perform additional checks to ensure exponents do not satisfy a certain algebraic property. In particular, the oracle checks that there is no sequence of exponents $e, e_1, \ldots, e_z$ such that $e = p_1 \cdot p_2 \cdots p_z$ and $p_i \mid e_i$ for all $i \in [z]$. First, we note that the primes $p_1, \ldots, p_z$ do not necessarily need to be distinct and e can be the product of prime powers. Furthermore, e is distinct from all of $e_1, \ldots, e_z$. However, the z exponents $e_1, \ldots, e_z$ do not need to be distinct. The exponent oracle continues to pick random exponents until this algebraic property does not hold. This ends up being important to avoid various attacks on multi-exponent one-more RSA assumptions (see Appendix H for details). We present this game in Figure 6 and the definition below.

Definition 4.4 ($(\epsilon, t, \ell, \ell')$-Chosen-Target, Algebraically-Restricted-Exponent RSA Inversion Assumption). Let λ be the security parameter and consider the game $\mathrm{G}^{\text{CT-ARE-RSA}}_{\text{Gen},\mathcal{A}}(\lambda)$ in Figure 6. The $(\epsilon, t, \ell, \ell')$-chosen-target, algebraically-restricted-exponent RSA inversion assumption is true for Gen, if for any adversary $\mathcal{A}$ that runs in time t, makes ℓ decryption queries, and makes ℓ' exponent queries, $\Pr[\mathrm{G}^{\text{CT-ARE-RSA}}_{\text{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon$.

We relate this to the chosen-target, restricted-exponent game without the algebraic restrictions (Def. 4.3) in Appendix H.2.

Relation to Chosen-Target RSA Inversion Assumption. Clearly, the chosen-target, (algebraically-)restricted-exponent variant is a no weaker assumption than the original chosen-target variant (Definition 3.5) from [11]. Recall that, in the proof of Theorem 4.2, it was required to generate fake challenge exponents co-prime to $\phi(N)$. This can be easily done when N is a strong RSA modulus. For this reduction, we would need to additionally be able to produce decryptions with respect to the fake challenge exponents. Unfortunately, this seems quite challenging. For example, suppose an adversary tried to produce pairs (e', d') such that $e'd' = 1 \bmod \phi(N)$. Then, the adversary could submit e' as a challenge exponent and use d' to answer decryption queries. It is well known that producing a pair (e', d') satisfying this property is equivalent to factoring N and, thus, breaking any RSA assumption immediately. This seems like a very challenging task for an adversary without knowledge of $\phi(N)$. We leave it as an open problem to determine their relationship.

Winning Condition. In some prior works, the adversary's winning condition is outputting a pair (X, Y) for e that is different from any query to O^{RSA}. In Appendix B, we show this is equivalent to our condition of outputting the tuple $(X_i, Y_i)_{i\in[z]}$ where $z > \text{cnt}_e$.

4.3 Connections to Strong RSA Assumption

Finally, we make connections between the above multi-exponent RSA assumptions and the strong RSA assumption in [10, 31]. We emphasize that the strong RSA assumptions in this section will not be used for proving security of any of our protocols.

Game $\mathrm{G}^{\text{CT-ARE-RSA}}_{\text{Gen},\mathcal{A}}(\lambda)$:	Oracle $O^{\text{RSA}}(X, e)$:	Oracle $O^{\mathcal{X}}()$:	Oracle $O^{\text{exp}}()$:
$(N, \mathcal{D}_N, (p, q)) \leftarrow_R \text{Gen}(1^\lambda)$ $\mathcal{S}_{\mathcal{X}} \leftarrow \emptyset, \mathcal{S}_{\text{exp}} \leftarrow \emptyset$ $\phi(N) \leftarrow (p-1)(q-1)$ $\text{cnt}_e \leftarrow 0, \forall e \in \mathbb{Z}^*_{\phi(N)}$ $e, (X_i, Y_i)_{i\in[z]} \leftarrow \mathcal{A}^{O^{\text{RSA}}, O^{\mathcal{X}}, O^{\text{exp}}}(N)$ **Return** 1 if and only if all the following hold: - $\text{cnt}_e < z$ - $e \in \mathcal{S}_{\text{exp}}$ - $\forall i \neq j \in [z], X_i \neq X_j$ - $\forall i \in [z], X_i \in \mathcal{S}_{\mathcal{X}}$ - $\forall i \in [z], Y_i^e = X_i \bmod N$	If $e \notin \mathcal{S}_{\text{exp}}$: **Return** $\perp$ $\text{cnt}_e \leftarrow \text{cnt}_e + 1$ $d \leftarrow e^{-1} \bmod \phi(N)$ $Y \leftarrow X^d \bmod N$ **Return** Y	$X \leftarrow_R \mathbb{Z}^*_N$ $\mathcal{S}_{\mathcal{X}} \leftarrow \mathcal{S}_{\mathcal{X}} \cup \{X\}$ **Return** X	$e \leftarrow_R \mathcal{D}_N$ $\mathcal{S}_{\text{exp}} \leftarrow \mathcal{S}_{\text{exp}} \cup \{e\}$ For $e' \in \mathcal{S}_{\text{exp}}$: $x \leftarrow e'$ For $e'' \in \mathcal{S}_{\text{exp}} \setminus \{e'\}$: $y \leftarrow \text{GCD}(x, e'')$ While $y > 1$: $x \leftarrow x/y$ $y \leftarrow \text{GCD}(x, e'')$ If $x = 1$: $\mathcal{S}_{\text{exp}} \leftarrow \mathcal{S}_{\text{exp}} \setminus \{e\}$ Go back to first step of O^{exp} to re-sample e. **Return** e

Figure 6: Chosen-Target, Restricted-Exponent RSA Inversion Game with all differences with the chosen-target RSA inversion game are highlighted in blue and ignoring highlighted parts in red. If you add the portions highlighted in red as well, we obtain the Chosen-Target, Algebraically-Restricted-Exponent RSA Inversion Game.

Strong RSA Assumption. In comparison with the standard RSA assumption, the strong RSA assumption provides more flexibility to the adversary to choose its own public exponent. The strong RSA problem states that the original RSA problem is intractable even when $\mathcal{A}$ is allowed to choose the public exponent $e \geq 3$. More specifically, given a RSA modulus N, and a random target X, it is infeasible to find any pair (Y, e) such that $X = Y^e \bmod N$ and $e \geq 3$.

Extending towards One-More Strong RSA Type Assumptions. A natural next step is to extend the strong RSA assumption using the one-more style definitions. To our knowledge, we are unaware of prior work that has formally defined this notion. We will present two natural definitions and show that they can be broken by an adversary. Before presenting these definitions, we quickly overview why the prior chosen-target, restricted-exponent RSA assumption (Definition 4.3) from Section 4.2 seems weaker. Recall that, in the strong RSA assumption, the adversary is able to choose any exponent $e \geq 3$. In contrast, for $\mathrm{G}^{\text{CT-RE-RSA}}_{\text{Gen},\mathcal{A},\ell}(\lambda)$, the output exponent must be one of the outputs of the challenge oracle O^{exp} and the adversary is not able to submit any decryption requests for exponents except for those output by O^{exp}. Therefore, it seems that the chosen-target, restricted-exponent assumption is weaker and not a truly natural extension of the strong RSA assumption. In Appendix D, we explore two variants where the adversary may choose exponents for decryption. We show that, in either case, there exists a polynomial time adversary breaking both assumptions.

5 RSA Signatures with Public Metadata

We start with RSA (non-blind) signatures with public metadata and show our modification does not degrade concrete security compared to the original protocol. We present this as further evidence that our modification does not degrade security substantially.

5.1 Preliminaries and Building Blocks

Unique Concatenation. We will utilize concatenation of multiple values where each unique tuple should result in a unique concatenation. That is, for any pair of tuples, (A, B) and (A', B'), it must be that their concatenation $A \,||\, B$ is identical to $A' \,||\, B'$ if and only if $A = A'$ and $B = B'$. This could also extend to tuples of size larger than two. A straightforward way to do this is to encode tuples (A, B) as $|A| \,||\, A \,||\, B$ where $|A|$ is the length of A stored in some fixed-length integer (such as 64 bits).

Message Encoding. In our protocols, we will utilize standard encoding algorithms for messages used in RSA signatures. Formally, we will assume that the message encoding consists of two algorithms: $H_{\mathcal{M}}$ and $\text{Verify}_{\mathcal{M}}$. The hash function $H_{\mathcal{M}}(X) : \{0,1\}^* \rightarrow \mathbb{Z}^*_N$ maps strings to elements in the multiplicative group modulo the RSA modulus N. The verification algorithm $\text{Verify}_{\mathcal{M}}(X, Y) : \{0,1\}^* \times \mathbb{Z}^*_N \rightarrow \{0,1\}$ checks and returns 1 if and only if Y corresponds to the output of $H_{\mathcal{M}}(X)$. In our work, we will consider two different encodings: full domain hash (FDH) and probabilistic signature scheme (PSS). At a high level, FDH is a determinsitic encoding algorithm that maps each message to a random element. In contrast, PSS is a randomized encoding algorithm that will map the same message to different outputs when evoked multiple times. The concrete security of PSS encodings was shown to be better than FDH encodings when applied to standard RSA signatures in [12]. In our implementations, we use PSS encodings due to its superior concrete security and randomized properties. Furthermore, we wish to align our implementation with current IRTF specifications for standard RSA blind signatures [29]. The parameters of PSS encodings that we use are from prior specifications [42] that essentially use a variant of PSS encodings with message recovery presented in [12]. For our proofs, we will prove security of our schemes assuming the usage of the FDH encoding. We chose this approach as to not over-complicate our proof with PSS encoding techniques that are directly borrowed from prior work [12]. Although, one can directly apply the proof techniques for PSS encodings from [12] to our security proofs. As we will use FDH, we present it formally.

Definition 5.1 (Full Domain Hash). The full domain hash (FDH) message encoding consists of two deterministic algorithms $H_{\mathcal{M}} : \{0,1\}^* \rightarrow \mathbb{Z}^*_N$ and $\text{Verify}_{\mathcal{M}} : \{0,1\}^* \times \mathbb{Z}^*_N \rightarrow \{0,1\}$ as follows:

- For string X, $H_{\mathcal{M}}(X)$ is a random element from $\mathbb{Z}^*_N$.
- For string X, $\text{Verify}_{\mathcal{M}}(X, Y) = 1$ if and only if $Y = H_{\mathcal{M}}(X)$.

Throughout the rest of this section and our proofs, we will assume that all hash functions are modeled as random oracles.

5.2 Our Protocol

For the reader's convenience, we quickly summarize some terimnology. We will denote the strong RSA modulus by N of bit length 2κ such that the two safe primes p, q (factors of N) are both of bit length κ. The public key will be $\text{pk} \leftarrow N$ which is the strong RSA modulus. The private key will be $\text{sk} \leftarrow \phi(N)$. As $p = 2p' + 1$ and

$q = 2q' + 1$ are safe primes, we know that both p' and q' are also primes. Therefore, $\phi(N) = (p-1)\cdot(q-1) = 4p'q'$.

Public Metadata Hash Function. Next, we define our hash function that is used to map public metadata to values in the group of RSA exponents $\mathbb{Z}^*_{\phi(N)}$. Recall that these are all elements that are invertible modulo $\phi(N)$. This is a straightforward task if $\phi(N)$ was publicly known by all parties. However, it is critical that $\phi(N)$ is hidden to all parties except the signer. Therefore, we must come up with a way to perform this mapping for users that do not know the value of $\phi(N)$. We present a simple way to do this that relies on the structure of $\mathbb{Z}^*_{\phi(N)}$ and the fact that N is a strong RSA modulus using a random hash function G.

$H_{\mathsf{MD}}(D)$:

(1) Compute $G(D)$ of length γ bits.
(2) Return $2 \cdot G(D) + 1$.

The above function simply returns a random odd number of length $\gamma + 1$ bits. Picking γ correctly, we can guarantee that the output of H_{MD} will always be smaller than the prime values p' and q'. This turns out to be sufficient to guarantee that outputs of H_{MD} will always be elements from $\mathbb{Z}^*_{\phi(N)}$.

Lemma 5.2. *Suppose N is a strong RSA modulus such that $N = pq$ where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes each of length κ and $\gamma \leq \kappa - 3$. For all $D \in \{0,1\}^*$, $H_{\mathsf{MD}}(D)$ is co-prime to $\phi(N)$.*

Proof. Fix any $x \in \{0,1\}^*$. First, we note that the output of $H_{\mathsf{MD}}(x)$ is always odd. Therefore, we know that if $H_{\mathsf{MD}}(x)$ is not co-prime to $\phi(N)$, then it must be that either $p' \mid H_{\mathsf{MD}}(x)$ or $q' \mid H_{\mathsf{MD}}(x)$. We know that the bit length of p' and q' is at least $\kappa - 1$ given that $p = 2p' + 1$ and $q = 2q' + 1$ and both p and q are κ bit long primes. Also note that the bit length of $H_{\mathsf{MD}}(x)$ is $\gamma + 1 \leq \kappa - 2$. Thus, we know that $H_{\mathsf{MD}}(x) < p'$ and $H_{\mathsf{MD}}(x) < q'$. Therefore, $H_{\mathsf{MD}}(x)$ is always co-prime to $\phi(N)$. □

RSA Signatures with Public Metadata. Using the public metadata hash function H_{MD}, we are ready to present our RSA signatures with public metadata. We assume that $H_{\mathcal{M}}$ and $\mathsf{Verify}_{\mathcal{M}}$ correspond to some message encoding algorithm (such as FDH in Definition 5.1). The Setup algorithm is identical to the standard RSA signatures without public metadata protocols except that in our Setup algorithm we only accept N if it is a strong RSA modulus. We present the formal algorithms for all of the necessary algorithms below.

$\mathsf{RSA}_{\mathsf{MD}}.\mathsf{Setup}(1^\lambda)$:

(1) Compute $(N, p, q) \leftarrow_R \mathsf{Gen}(1^\lambda)$ and $\phi(N) \leftarrow (p-1)(q-1)$.
(2) Return $\mathsf{pk} \leftarrow N, \mathsf{sk} \leftarrow \phi(N)$.

$\mathsf{RSA}_{\mathsf{MD}}.\mathsf{Sign}(M, D, \mathsf{pk} \leftarrow N, \mathsf{sk} \leftarrow \phi(N))$:

(1) Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$ and $d_{\mathsf{MD}} \leftarrow (e_{\mathsf{MD}})^{-1} \bmod \phi(N)$.
(2) Compute $M_{\mathsf{MD}} \leftarrow H_{\mathcal{M}}(M \mid\mid D)$.
(3) Return signature $S \leftarrow (M_{\mathsf{MD}})^{d_{\mathsf{MD}}} \bmod N$.

$\mathsf{RSA}_{\mathsf{MD}}.\mathsf{Verify}(S, M, D, \mathsf{pk} \leftarrow N)$:

(1) Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$ and $X \leftarrow S^{e_{\mathsf{MD}}} \bmod N$.
(2) Return $\mathsf{Verify}_{\mathcal{M}}(M \mid\mid D, X)$.

Correctness. First, we show that $d_{\mathsf{MD}} \leftarrow (e_{\mathsf{MD}})^{-1} \bmod \phi(N)$ is actually computable. We will rely on Lemma 5.2 stating that all outputs of H_{MD} are co-prime with $\phi(N)$ and the signing algorithm is always well-defined. Next, we will show that the verification succeeds for well-formed signatures. Let S be a signature that is produced by following the Setup and Sign algorithms properly for input message M and public metadata D. Then, we know that the signature satisfies: $S = (H_{\mathcal{M}}(M \mid\mid D))^{1/H_{\mathsf{MD}}(D)} \bmod N$. Then, the verification algorithm will return the following result:

$$\mathsf{Verify}_{\mathcal{M}}(M \mid\mid D, S^{\cdot H_{\mathsf{MD}}(D)}) = \mathsf{Verify}_{\mathcal{M}}(M \mid\mid D, H_{\mathcal{M}}(M \mid\mid D)) = 1$$

as $S^{H_{\mathsf{MD}}(D)} = H_{\mathcal{M}}(M \mid\mid D)$ and $\mathsf{Verify}_{\mathcal{M}}(X, H_{\mathcal{M}}(X)) = 1$.

Unforgeability. Finally, we show that the concrete security bounds for unforgeability are similar to those proven by [12] for standard RSA signatures with FDH message encodings except for a multiplicative factor loss in number of hashing oracle calls to H_{MD}. At a high level, our security proof will construct an adversary $\mathcal{A}$ for the multi-exponent RSA game given exponents from $[3, 2^{\kappa-2}]$. Afterwards, we can apply Theorem 4.2 to obtain security based on the standard RSA assumption. Due to lack of space, we present the formal theorem here and defer the full proof to Appendix E.

Theorem 5.3. *Suppose that $(H_{\mathcal{M}}, \mathsf{Verify}_{\mathcal{M}})$ correspond to full domain hash (FDH) message encoding. Assuming the (ϵ, t)-RSA assumption (Definition 3.3) with exponents in $[3, 2^{\kappa-2}]$ and the random oracle model, then $\mathsf{RSA}_{\mathsf{MD}}$ is $(\epsilon_F, t_F, \ell_F)$-strong one-more unforgeable (Definition 2.2) where*

- $t_F = t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$
- $\epsilon_F = q_{\mathsf{MD}} \cdot q_{\mathcal{M}} \cdot \epsilon$

and the adversary $\mathcal{A}$ runs in time at most t_F and makes at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle and hash functions $H_{\mathcal{M}}$ and H_{MD} respectively.

The concrete security is almost identical as those proven in [12] for RSA signatures except for a multiplicative q_{MD} loss in ϵ. In Appendix F, we present a modification with identical bounds to [12].

Discussion on Multi-Exponent Attacks. In Section 7, we outline multi-exponent attacks that assume the availability of an oracle for RSA decryption of arbitrary elements. Those attacks are not applicable in this setting since the signer receives a message M and metadata D and signs (i.e., RSA decryption) on $H_{\mathcal{M}}(M \mid\mid D)$. If an attacker wishes to RSA decrypt an element X with respect to exponent $e = H_{\mathsf{MD}}(D)$, it must find an input message M such that $H_{\mathcal{M}}(M \mid\mid D) = X$. This makes the attacks intractable for our non-blind RSA signatures. See Appendix E.1 for more details.

6 RSA Blind Signatures with Public Metadata

We present our RSA blind signature with public metadata using the same public metadata hash function H_{MD} from Section 5.2. We note our scheme is a modified variant of the protocol presented in [3].

Choosing Public Metadata Set. We discuss key requirements for choosing the public metadata set **MD**. It will be critical that the set **MD** is chosen ahead of time and not too large for both anonymity and unforgeability. Recall that we ensure anonymity (unlinkability) amongst all users with the same public metadata. In the extreme case when **MD** is too large, each user could be assigned their own public metadata completely eliminating anonymity.

Additionally, it is also critical that the set **MD** must be chosen ahead of time during the setup phase. Furthermore, the issuer should not sign any blinded messages for public metadata D outside of the permitted set $D \notin \mathbf{MD}$ fixed during the setup phase. These will be critical to avoid forgeability attacks (as we discuss later). The

restriction of choosing the public metadata set during setup does not impede most applications. In our two example applications, the public metadata set can be fixed to be all countries and all valid expiration timestamps between scheduled key rotations.

Our Protocol. We will assume some message encoding protocol defined by $H_{\mathcal{M}}$ and $\mathsf{Verify}_{\mathcal{M}}$. As a note, we will utilize a modification presented by Lysyanskaya [40] that enables providing unlinkability even against malicious signers. It was shown that appending a random string to the message of bit length η is sufficient to provide anonymity against even adversarially chosen RSA modulus. Additionally, we also need checks into the setup portion to ensure that exponents corresponding the public metdata set **MD** do not satisfy certain algebraic properties.

$\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Setup}(1^\lambda, \mathbf{MD})$:

1. Execute $(N, \phi(N)) \leftarrow \mathsf{RSA}_{\mathsf{MD}}.\mathsf{Setup}(1^\lambda)$.
2. Generate random string $\mathsf{salt} \in \{0,1\}^\lambda$.
3. For $D \in \mathbf{MD}$:
   (a) Compute $e \leftarrow H_{\mathsf{MD}}(\mathsf{salt} \,||\, D)$.
   (b) For $D' \in \mathbf{MD} \setminus \{D\}$:
      (i) Compute $e' \leftarrow H_{\mathsf{MD}}(\mathsf{salt} \,||\, D')$.
      (ii) Compute $g \leftarrow \mathsf{GCD}(e, e')$ as integers.
      (iii) While $g > 1$:
         (A) Set $e \leftarrow e/g$ as integers.
         (B) Compute $g \leftarrow \mathsf{GCD}(e, e')$ as integers.
      (iv) If $e = 1$, go back and repeat from Step 2.
4. Return $\mathsf{pk} \leftarrow (N, \mathsf{salt}), \mathsf{sk} \leftarrow \phi(N)$.

$\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Blind}(M, D, \mathsf{pk} \leftarrow (N, \mathsf{salt}))$:

1. If $D \notin \mathbf{MD}$, return $\perp$.
2. Pick random η-length string rand and set $M' \leftarrow M \,||\, \mathsf{rand}$.
3. Pick R uniformly at random from $\mathbb{Z}_N^*$ and set $\mathsf{st} \leftarrow (\mathsf{rand}, R)$.
4. Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(\mathsf{salt} \,||\, D)$.
5. Compute *blinded message* $B_M \leftarrow R^{e_{\mathsf{MD}}} \cdot H_{\mathcal{M}}(M' \,||\, D) \bmod N$.
6. Return (st, B_M).

$\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Sign}(B_M, D, \mathsf{pk} \leftarrow (N, \mathsf{salt}), \mathsf{sk} \leftarrow \phi(N))$:

1. If $D \notin \mathbf{MD}$, return $\perp$.
2. Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(\mathsf{salt} \,||\, D)$.
3. Compute $d_{\mathsf{MD}} \leftarrow (e_{\mathsf{MD}})^{-1} \bmod \phi(N)$.
4. Return $S' \leftarrow (B_M)^{d_{\mathsf{MD}}} \bmod N$.

$\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Finalize}(\mathsf{st} \leftarrow (\mathsf{rand}, R), S', M, D, \mathsf{pk} \leftarrow (N, \mathsf{salt}))$:

1. Compute $S \leftarrow S' \cdot R^{-1} \bmod N$.
2. If $\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Verify}((S, \mathsf{rand}), M, D, \mathsf{pk}) = 0$, return $\perp$.[2]
3. Return *signature* (S, rand).

$\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Verify}((S, \mathsf{rand}), M, D, \mathsf{pk} \leftarrow (N, \mathsf{salt}))$:

1. Compute $M' \leftarrow M \,||\, \mathsf{rand}$ and $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(\mathsf{salt} \,||\, D)$.
2. Compute $X \leftarrow S^{e_{\mathsf{MD}}} \bmod N$.
3. Return $\mathsf{Verify}_{\mathcal{M}}(M' \,||\, D, X)$.

Correctness. By Lemma 5.2, we know that the algorithm above is well-defined as the signer can always compute an inverse of $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$. To show that the above correctly verifies well-formed signatures, we consider an execution of the blind signing protocol. The output of Blind is $R^{e_{\mathsf{MD}}} \cdot H_{\mathcal{M}}(M' \,||\, D)$. The output of Sign is $(R^{e_{\mathsf{MD}}} \cdot H_{\mathcal{M}}(M' \,||\, D))^{(e_{\mathsf{MD}})^{-1}} = R \cdot H_{\mathcal{M}}(M' \,||\, D)^{(e_{\mathsf{MD}})^{-1}}$. Finally, the output of Finalize is $H_{\mathcal{M}}(M' \,||\, D)^{(e_{\mathsf{MD}})^{-1}}$. Then Verify will output $\mathsf{Verify}_{\mathcal{M}}(M' \,||\, D, H_{\mathcal{M}}(M' \,||\, D)^{(e_{\mathsf{MD}})^{-1} \cdot e_{\mathsf{MD}}}) = \mathsf{Verify}_{\mathcal{M}}(M' \,||\, D, H_{\mathcal{M}}(M' \,||\, D)) = 1$. Therefore, a well-formed signature will always be verified correctly.

Efficiency. We note that the efficiency is similar with only a couple differences. First, we perform RSA operations with a larger exponent than standard RSA signatures. There are additional calls to H_{MD} and the signer must perform an inversion. However, we show that this only incurs minimal overhead (see Section 8). The major difference is setup that requires generating strong RSA modulus and checking exponents requiring $O(|\mathbf{MD}|^2)$ time, but this is done once offline outside of the issuance and verification phases. We show in Section 7.1 that the number of salt resamples is very small for most choices of public metadata set sizes **MD**.

Our construction requires larger exponents than typically meaning more computation. In Appendix G, we discuss why restricting **MD** prevents potential DoS attacks leveraging large exponents.

6.1 Unforgeability

Next, we prove the unforgeability of $\mathsf{BlindRSA}_{\mathsf{MD}}$ by showing that any adversary that can forge signatures in $\mathsf{BlindRSA}_{\mathsf{MD}}$ is able to break the chosen-target, algebraically-restricted-exponent RSA inversion assumption. At a high level, we show that one can simulate signing oracle queries using the RSA decryption oracle.

Theorem 6.1. *Suppose that $(H_{\mathcal{M}}, \mathsf{Verify}_{\mathcal{M}})$ correspond to full domain hash (FDH) message encoding. Assuming the $(\epsilon, t, \ell, |\mathbf{MD}|)$-chosen-target, algebrically-restricted-exponent RSA inversion assumption (Definition 4.4) with exponents in $[3, e_{\mathbf{max}}]$ such that $e_{\mathbf{max}} = 2^{\kappa-2}$ and the random oracle model, then $\mathsf{BlindRSA}_{\mathsf{MD}}$ is $(\epsilon_F, t_F, \ell_F)$-strong one-more unforgeable (Definition 2.2) where*

- $\epsilon_F = \epsilon + 2^{-\kappa+2\log q_{\mathcal{M}}}$
- $t_F = t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$
- $\ell_F = \ell$

and the adversary $\mathcal{A}$ runs in time at most t_F and makes at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle and hash functions $H_{\mathcal{M}}$ and H_{MD} respectively.

Proof. To prove this, we will show that if there exists some adversary $\mathcal{A}_F$ that successfully forges signatures for $\mathsf{BlindRSA}_{\mathsf{MD}}$ to win the strong one-more unforgeability game, then we can use $\mathcal{A}_F$ to construct an adversary $\mathcal{A}$ to break the chosen-target, restricted-exponent RSA inversion assumption. The forger $\mathcal{A}_F$ runs in time t_F, wins the game with probability ϵ_F and uses at most ℓ_F signing queries. Our reduction will only increase the running time of the adversary $\mathcal{A}$ by a factor of $O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$, reduces the winning probability by at most $2^{-\kappa+2\log q_{\mathcal{M}}}$, maintains the same number of oracle queries between signing and RSA decryption and makes at most $|\mathbf{MD}|$ exponent oracle queries. Furthermore, the exponent distributions are identical as our setup performs similar checks.

Our adversary $\mathcal{A}$ will simulate $\mathcal{A}_F$ to break the chosen-target, restricted-exponent RSA game. Note that $\mathcal{A}$ needs to be able to successfully simulate all hash and signing oracle queries performed by $\mathcal{A}_F$. $\mathcal{A}$ simulates these queries as follows:

For any query to $H_{\mathsf{MD}}(D)$:

1. If $\mathbf{M}_{\mathsf{MD}}[D]$ is set, return $\mathbf{M}_{\mathsf{MD}}[D]$.
2. Compute $\mathbf{M}_{\mathsf{MD}}[D] \leftarrow \mathcal{O}^{\mathsf{exp}}()$ and return $\mathbf{M}_{\mathsf{MD}}[D]$.

[2] Optional for correctness but is done to check if the server used the correct key.

For any query to $H_{\mathcal{M}}(M, D)$:

(1) If $\mathbf{M}_{\mathcal{M}}[M, D]$ is set, return $\mathbf{M}_{\mathcal{M}}[M, D]$.
(2) Compute $\mathbf{M}_{\mathcal{M}}[M, D] \leftarrow O^{\mathcal{X}}()$ and return $\mathbf{M}_{\mathcal{M}}[M, D]$.

For any query to $O^{\mathsf{Sign}}(X, D)$:

(1) Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(\mathsf{salt} \,||\, D)$ and return $O^{\mathsf{RSA}}(X, e_{\mathsf{MD}})$.

Adversarial Advantage. First, we note that the simulated outputs of O^{exp} and the real outputs of H_{MD} are identical as $e_{\mathbf{max}} = 2^{\kappa-2}$. Similarly, $O^{\mathcal{X}}$ returns random elements identical to the FDH encoding. Finally, we note that signing always returns a valid signature as O^{RSA} only receives exponents that are output by O^{exp}. Finally, $\mathcal{A}_F$ will output the values $D, (S_i, M_i)_{i\in[x]}$ where $x \geq \mathsf{cnt}_D + 1$ such that cnt_D is the maximum number of signing oracles performed with public metadata D. If $\mathcal{A}_F$ successfully forges, then we know that $S_i^{H_{\mathsf{MD}}(D)} = H_{\mathcal{M}}(M_i \,||\, D)$ for all $i \in [x]$. Then, $\mathcal{A}$ will output $H_{\mathsf{MD}}(D), (S_i, H_{\mathcal{M}}(M_i \,||\, D))_{i\in[x]}$ as its output to the chosen-target, restricted-exponent RSA inversion game. Note, $\mathcal{A}$ wins the game as long as they are all valid decryptions and there are no collisions in the output of $H_{\mathcal{M}}(M_i \,||\, D)$. As there are at most $q_{\mathcal{M}}$ queries to $H_{\mathcal{M}}$, we know the collision probability is at most $q_{\mathcal{M}}^2/2^{-\kappa} = 2^{-\kappa+2\log q_{\mathcal{M}}}$ since $|\mathbb{Z}_N^*| \geq 2^{-\kappa}$. Therefore, we know that $\mathcal{A}$ wins the game if $\mathcal{A}_F$ wins the unforgeability game and there are no collisions in the outputs of $H_{\mathcal{M}}$. So, the probability that $\mathcal{A}$ wins is $\epsilon_F - 2^{-\kappa+2\log q_{\mathcal{M}}}$.
Adversarial Running Time. We know that $\mathcal{A}_F$ runs time at most t_F. For each hash and signing oracle query, $\mathcal{A}$ performs $O(1)$ additional operations to correctly simulate responses. As there are most $q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F$ queries overall, we see that $\mathcal{A}$ requires $O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$ additional time. Furthermore, we note that each signing query results in exactly one decryption oracle query. Therefore, we know that at most ℓ_F decryption oracle queries are made. Finally, each unique metadata $D \in \mathbf{MD}$ results in at most one exponent query meaning at most $|\mathbf{MD}|$ exponent queries. □

Discussion about Exponent Setup Check. In the setup, we sample random salts until the generated exponents do not satisfy a certain algebraic property. In particular, we check that there does not exist some metadata $D \in \mathbf{MD}$ where $e = H_{\mathsf{MD}}(\mathsf{salt} \,||\, D) = p_1 \cdot p_2 \cdots p_z$ is the product of z primes that are not necessarily distinct (so, e can be the product of prime powers). Additionally, there must exist (not necessarily distinct) metadata $D_1, \ldots, D_z \in \mathbf{MD} \setminus \{D\}$ such that $p_i \mid H_{\mathsf{MD}}(\mathsf{salt} \,||\, D_i)$. We perform this check as there are forgery attacks [16, 22] if an adversary can find such a sequence $D, D_1, \ldots, D_z$ of public metadata (see Appendix H for full attack details). Our requirement that the issuer only operates over valid metadata $D \in \mathbf{MD}$ ensures that the adversary is restricted to signing oracle (RSA inversion oracle) queries in the set $\{H_{\mathsf{MD}}(\mathsf{salt} \,||\, D) : D \in \mathbf{MD}\}$. We show these checks fail with small probability in Appendix H meaning salts are not re-sampled often.
Omitting Exponent Checks. It is possible to omit the exponent checks during setup if one is willing to withstand a small probability of a forgery. In Section 7.1, we analyze the probability that a random salt satisfies the algebraic properties necessary for a forging attack. If the public metadata set $\mathbf{MD}$ is not too large, we show this probability is very small (see Figure 7). In our country example where there $|\mathbf{MD}| \leq 200$ countries, the probability that the forging attack in [16] succeeds is at most 2^{-52} and 2^{-72} for RSA modulus of length 2048 and 3072 respectively. Omitting exponent checks means that unforgeability probabilistically reduces to Def. 4.4 (see Section 7.1 for probabilities). We note that omitting exponent checks is identical to directly reducing to the chosen-target, restricted-exponent game (without the checks in the exponent oracle) in Def. 4.3. We formalize this relationship in Appendix H.2.

6.2 Unlinkability

To prove that $\mathsf{BlindRSA}_{\mathsf{MD}}$ provides unlinkability, we will rely on prior results in [40] that show that appending random strings to the message is sufficient to provide protections against maliciously generated parameters. At a high level, the work in [40] shows that for any maliciously chosen RSA modulus N and public exponent e, unlinkability for standard RSA blind signatures with appended random strings of bit length η holds with probability except $t \cdot 2^{-\eta}$ for adversaries running in time t. For our case, the adversary is able to choose the public metadata D. However, this ends up being a more restrictive way for the adversary to choose as the final exponent $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$ as opposed to choosing e_{MD} directly.

Theorem 6.2. *Assuming the random oracle model,* $\mathsf{BlindRSA}_{\mathsf{MD}}$ *satisfies* $(t \cdot 2^{-\eta}, t)$*-unlinkability (Definition 2.3).*

Proof. Consider the unlinkability games for $\mathsf{BlindRSA}_{\mathsf{MD}}$ with public metadata and the standard RSA blind signatures in [40]. There is only one difference between the two settings. In the former, the adversary must choose D. The final RSA public exponent becomes $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$. In the latter, the adversary is able to directly choose the final public exponent. Note, we can do this because $\mathsf{BlindRSA}_{\mathsf{MD}}$ is nearly identical to standard RSA blind signatures after augmenting the public exponent to be $H_{\mathsf{MD}}(D)$. Clearly, the scenarios studied in [40] are more favorable for the adversary. Now, suppose there exists an adversary $\mathcal{A}$ for the unlinkability of $\mathsf{BlindRSA}_{\mathsf{MD}}$ with probability ϵ and running time t. $\mathcal{A}$ outputs values public key N, messages M_0 and M_1 and public metadata D in the first step. We construct $\mathcal{A}'$ that instead outputs $e = H_{\mathsf{MD}}(D)$ as the public exponent. Then, $\mathcal{A}'$ also wins with probability ϵ and time t. □

7 Analyzing Multi-Exponent Attacks

First, we note our public metadata hash function H_{MD} is similar to division intractable hash functions used in [32] and analyzed in [22]. At a high level, these attacks work by trying to obtain a chain of exponents $e, e_1, \ldots, e_z$ with the following properties. Suppose $e = p_1 \cdot p_2 \cdots p_z$ is the product of z primes that are not necessarily distinct (in other words, e can be the product of prime powers). First, e is distinct from all of $e_1, \ldots, e_z$. However, we note that the remaining z exponents, $e_1, \ldots, e_z$, do not need to be distinct and can repeat the same exponent. Secondly, p_i divides e_i evenly as an integer, $p_i \mid e_i$. Borisov [16] showed that a forging attack is possible in this case by leveraging similar algebraic properties (this could also be translated to our chosen-target, restricted-exponent RSA inversion assumption in Definition 4.3 as well). We present this attack below. We also note that it is similar to the adversaries we presented for our chosen-exponent RSA assumptions in Appendix D.

The attack by Borisov [16] aims to find a sequence of public metadata $D, D_1, \ldots D_z$ with the following properties. Let $e = H_{\mathsf{MD}}(\mathsf{salt} \,||\, D) = p_1 \cdot p_2 \cdots p_z$ that is the product of z (not necessarily distinct)

primes. Then, p_i should divide $e_i = H_{\mathsf{MD}}(\text{salt} \mid\mid D_i)$. That is, $p_i \mid e_i$. Then, the forging adversary does the following:

(1) Set $x_0 = H(M \mid\mid D)$.
(2) For each $i \in [z]$:
   (a) Submit $x_{i-1}^{e_i/p_i}$ for blind signing with D_i and obtain x_i.
(3) Return forgery x_z for message M and public metadata D.

Each intermediate blind signing output for every $i \geq 1$ satisfies:

$$x_i = H(M \mid\mid D)^{(e_1/p_1)\cdots(e_i/p_i)(1/e_1)\cdots(1/e_i)} = H(M \mid\mid D)^{1/(p_1\cdots p_i)}$$

Then, $x_z = H(M \mid\mid D)^{1/(p_1\cdots p_z)} = H(M \mid\mid D)^{1/e}$ is the forgery.

7.1 Analyzing Exponent Check

In our setup phase, we run exponent checks to guarantee that this algebraic property does not exist amongst valid public metadata in the set **MD**. In this section, we analyze the existence of such bad sequences for a random choice of salt assuming H_{MD} uses a random oracle. This analysis will be useful in two scenarios. First, it bounds the number of salt re-samples during the setup phase. Secondly, in the case that the exponent check is omitted, this upper bounds the probability that the forging attack in [16] has the necessary algebraic property as a function of κ (prime bit-length) and $|\mathbf{MD}|$.

Theorem 7.1. *Consider security parameter λ and public metadata set* **MD**. *Suppose that H_{MD} outputs γ-bit odd integers where $\gamma \leq \kappa - 2$. Let E be the event that there exists $D, D_1, \ldots, D_z \in \mathbf{MD}$ such that:*

(1) *$D \neq D_i$ for all $i \geq 1$, but the remaining z metadata, $D_1, \ldots, D_z$, are not necessarily distinct;*
(2) *$e = H_{\mathsf{MD}}(\text{salt} \mid\mid D) = p_1 \cdot p_2 \cdots p_z$ where each p_i is a prime number that are not necessarily distinct;*
(3) *$e \mid e_i = H_{\mathsf{MD}}(\text{salt} \mid\mid D_i)$ for all $i \geq 1$.*

Then, for any randomly chosen salt and for every integer $x \geq 2$, $\Pr[E] \leq \frac{|\mathbf{MD}|^2}{x} + 2|\mathbf{MD}| \cdot \rho(\gamma/\log x)$ *where ρ is Dickman's function.*

Proof. In our analysis, we will rely on Dickman's function ρ for the analysis of smooth numbers. In particular, the probability that a random odd integer from $[1, y]$ has only prime factors no larger than x is $2 \cdot \rho(\log y/\log x)$. Let E' be the event that there exists any $D \in \mathbf{MD}$ such that $H_{\mathsf{MD}}(\text{salt} \mid\mid D)$ has no prime factors larger than x. Then, we know that $\Pr[E'] \leq 2|\mathbf{MD}| \cdot \rho(\gamma/\log x)$ since H_{MD} outputs γ-bit odd integers. Suppose that event E' does not occur. Fix any $D \in \mathbf{MD}$ and we want to bound the probability that there exists some sequence starting with D satisfying event E. Pick any prime factor p of $H_{\mathsf{MD}}(\text{salt} \mid\mid D)$ where $p > x$. If D is the start of the sequence satisfying the event E, then it better be that there exists some metadata $D' \in \mathbf{MD}$ such that p divides $H_{\mathsf{MD}}(\text{salt} \mid\mid D')$. The probability that a prime p divides a random odd integer from $[3, 2^{\kappa-2}]$ (output of H_{MD} for some $D' \in \mathbf{MD}$) is $1/p < 1/x$. Then, the probability that there exists any $D' \in \mathbf{MD}$ such that p divides $H_{\mathsf{MD}}(\text{salt} \mid\mid D')$ is at most $|\mathbf{MD}|/x$. In other words, the probability D is the start of some sequence satisfying event E is at most $|\mathbf{MD}|/x$. Using a Union bound over D, we get that $\Pr[E \mid \neg E'] \leq |\mathbf{MD}|^2/x$. Putting it altogether, we get the following: $\Pr[E] \leq \Pr[E \mid \neg E'] + \Pr[E'] \leq |\mathbf{MD}|^2/x + 2|\mathbf{MD}| \cdot \rho(\gamma/\log x)$. □

To use the above, one should attempt to find an integer x that minimizes the probability $\Pr[E]$. We do this for different κ with $\gamma = \kappa - 3$ and $|\mathbf{MD}|$ in Figure 7. In general, the probabilities are very small meaning that the expected number of times that salt needs to be re-sampled is very small (salts are almost never re-sampled). We also prove an asymptotic bound (see Appendix H.1) on the above probability, but they are much looser than the values in Figure 7.

Public Metadata Set Size ($|\mathbf{MD}|$)	$\kappa = 1024$	$\kappa = 1536$	$\kappa = 2048$
100	2^{-54}	2^{-72}	2^{-87}
200	2^{-52}	2^{-70}	2^{-85}
500	2^{-50}	2^{-68}	2^{-83}
1,000	2^{-49}	2^{-66}	2^{-82}
5,000	2^{-45}	2^{-63}	2^{-78}
10,000	2^{-43}	2^{-61}	2^{-76}
50,000	2^{-40}	2^{-58}	2^{-73}
100,000	2^{-38}	2^{-56}	2^{-71}
1,000,000	2^{-33}	2^{-51}	2^{-66}

Figure 7: Table with upper bounds of $\Pr[E]$ from Theorem 7.1 with $\gamma = \kappa - 3$ and varying sizes of $|\mathbf{MD}|$. Recall that the RSA modulus bit-length is 2κ.

Modulus Bit Length	**Algorithm**	**Public Metadata**	**No Public Metadata**
2048	Blind	1.4 ± 0.01	0.31 ± 0.01
2048	Sign	4.3 ± 0.01	1.6 ± 0.01
2048	Finalize	1.1 ± 0.01	0.028 ± 0.001
2048	Verify	1.1 ± 0.01	0.025 ± 0.001
3072	Blind	4.1 ± 0.11	0.61 ± 0.01
3072	Sign	12 ± 0.16	4.1 ± 0.01
3072	Finalize	3.4 ± 0.01	0.053 ± 0.001
3072	Verify	3.4 ± 0.01	0.053 ± 0.001
4096	Blind	8.9 ± 0.12	1.2 ± 0.01
4096	Sign	26 ± 0.16	8.1 ± 0.01
4096	Finalize	7.9 ± 0.01	0.088 ± 0.001
4096	Verify	7.9 ± 0.01	0.088 ± 0.001

Figure 8: Comparison of RSA blind signature with and without public metadata. Computational costs are presented in milliseconds. Each experiment was repeated 100 times.

8 Experimental Evaluation

Our Implementation. Our implementations of RSA blind signatures with and without public metadata may be found in [7]. Both implementations utilize PSS message encodings and rely on BoringSSL [33] for cryptographic primitives. For our protocol, the random string for unlinkability will be $\eta = 384$ bits following [40, 42] along with SHA384 and 384-bit salts for our hash function.

Experimental Setup. We conducted our experiments using Ubuntu PCs with 4 cores, AMD EPYC 7B12 2.2 GHz and 32 GB of RAM. Our experiments are all executed using a single thread.

Key Generation. For a standard RSA key generation protocol that does not support public metadata, key generation times range from 0.09 to 1 second for RSA modulus sizes from 2048 to 4096 bits. For our protocol the key needs to have a strong RSA modulus, RSA key generation takes around 2 seconds for 2048, 45 seconds for 3072 and 90 seconds for 4096 bit strong moduli. We present the computational cost of our exponent checks in Figure 10. As setup is an infrequent operation, the increase in setup time is inconsequential.

Comparison with RSA Blind Signatures. We report all our results of the computational costs in Figure 8. Note that Verify is also

	[56]	[60]	Ours (2048)	Ours (3072)
Blind	1.6 ± 0.08	1.1 ± 0.13	1.4 ± 0.01	4.1 ± 0.11
Sign	1.0 ± 0.03	5.6 ± 0.38	4.3 ± 0.01	12 ± 0.16
Finalize	3.3 ± 0.21	6.6 ± 0.47	1.1 ± 0.01	3.4 ± 0.01
Verify	3.9 ± 0.28	1.3 ± 0.05	1.1 ± 0.01	3.4 ± 0.01
Signature Size	48	145	256	384
Security Level	128	192	112	128

Figure 9: Comparison of $\mathsf{BlindRSA}_{\mathsf{MD}}$ with 2048-bit and 3072-bit modulus and solutions using pairings [56] and POPRFs with only secret key verification [60]. Computational costs are presented in milliseconds and signature sizes in bytes. Security levels are presented for the underlying curve or RSA group. Each experiment was repeated 100 times.

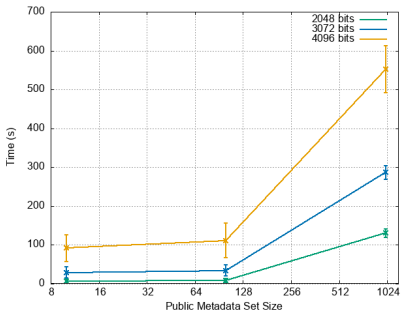

Figure 10: Setup time for different sizes of moduli against different Public Metadata set sizes. Every experiment was repeated 10 times.

a sub-routine for the Finalize routine. In general, we note that the protocol with public metadata is slower. This is expected as Sign requires an additional inverse operation modulo $\phi(N)$ to compute the private exponent. Moreover, the output of H_{MD} and hence the public exponent can be quite large that is needed for security. Therefore, it is not surprising that the public metadata variant is slower. Nevertheless, the RSA blind signatures with public metadata are still more than sufficiently fast enough for real-world applications. Both protocols have identical signature sizes depending on modulus length.

Comparison with Pairing Variant. We compare with the public metadata solution in [56] using pairings in Figure 9. We implemented the protocol in C++ using the curve BLS12-381 with the AMCL library [59] following the IRTF draft [14]. We compare with our 2048-bit modulus solution here following the draft [6] and with 3072-bit modulus solution in Figure 9. Our scheme was similar during Blind, 4x slower during Sign but 3x faster during Finalize. Our scheme is also 3.5x faster for Verify. The signature size for this protocol is 5x smaller than ours. We note again that pairings are not readily available in production cryptographic libraries.

Comparison with OPRF Variant. We compare with OPRF-based protocols with public metadata [36, 60] that only support secret key verification in Figure 9. We used the implementation available in Rust [49] with the curve P384 following the RFC [26] and compare with our 2048-bit modulus protocol here following the draft [6] and with 3072-bit modulus solution in Figure 9. Our scheme was slightly slower during Blind, slightly faster for Sign, Verify and 6x faster for Finalize. This protocol's signature size is smaller than our schemes.

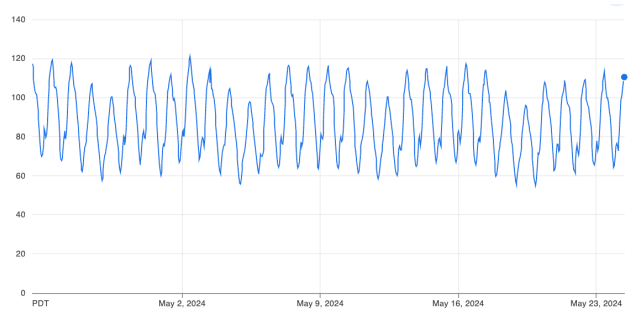

Figure 11: Queries per second measured every minute period.

9 Deployment Telemetry

We report anonymous telemetry over 30 days from April 25, 2024 until May 23, 2024 on our deployment of RSA blind signatures with public metadata with 2048-bit modulus for GoogleOne VPN.

Query Volume. We recorded query volume over each minute period in Figure 11. Our systems served 5,953 queries on average every minute. At peak, there were 7,276 queries within one minute. This works out to 99.2 queries per second on average and 121.3 queries per second at peak. On average, we received more than 85 million queries per day and more than 250 million total queries over the entire month. Queries were split 50.6% for issuance and the remainder for redemption. Issuance is naturally higher as some issued tokens may never be redeemed. The diurnal query pattern reflects the concentration of users in North America and Europe.

Latency. Overall, the median latency is stable for both operations at 33 ms for issuance and 91 ms for redemption. We also report that the 90-th (99-th) percentile latencies were 48 ms (57 ms) and 240 ms (352 ms) for issuance and redemption respectively. The higher latency (and variance) of redemption come from database reads and write for tracking and preventing double spending of tokens.

Cost Modeling. For our deployment, the majority of our costs are tied to computational costs and long-term storage for double spending checks. The cost of storing all spent tokens over the month cost less than $400 (or less than $0.16 every 100,000 redeemed tokens). Computational costs were less than $2300 (or less than $0.92 every 100,000 queries). All costs estimated using standard Google cloud computing and database pricing found here [34].

10 Conclusions

In this paper, we present a protocol for RSA blind signatures that support public metadata. We prove our schemes secure and provide strong evidence that concrete security of our scheme is nearly identical to standard RSA blind signatures. Furthermore, we show that our public metadata protocol is concretely efficient with comparable overhead as standard RSA blind signatures and other solutions using less available cryptography or without public key verification. We also report on anonymous deployment telemetry to showcase the scalability of our protocol in real-world settings. We leave the following open question: Can we prove security of our protocol from the chosen-target RSA inversion assumption (Def. 3.5)?

References

[1] Michel Abdalla, Chanathip Namprempre, and Gregory Neven. 2006. On the (Im)possibility of Blind Message Authentication Codes. In *CT-RSA 2006 (LNCS, Vol. 3860)*, David Pointcheval (Ed.). Springer, Heidelberg, Berlin, Heidelberg, 262–279.

[2] Masayuki Abe and Jan Camenisch. 1997. Partially Blind Signature Schemes. In *1997 Symposium on Cryptography and Information Security*.

[3] Masayuki Abe and Eiichiro Fujisaki. 1996. How to Date Blind Signatures. In *ASIACRYPT'96 (LNCS, Vol. 1163)*, Kwangjo Kim and Tsutomu Matsumoto (Eds.). Springer, Heidelberg, Kyongju, Korea, 244–251. https://doi.org/10.1007/BFb0034851

[4] Masayuki Abe and Tatsuaki Okamoto. 2000. Provably Secure Partially Blind Signatures. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 1880)*. Springer, 271–286. https://doi.org/10.1007/3-540-44598-6_17

[5] Martin R Albrecht, Alex Davidson, Amit Deo, and Daniel Gardham. 2024. Crypto Dark Matter on the Torus: Oblivious PRFs from Shallow PRFs and TFHE. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 447–476.

[6] Ghous Amjad, Scott Hendrickson, Christopher Wood, and Kevin Yeo. 2023. Partially Blind RSA Signatures. https://datatracker.ietf.org/doc/draft-amjad-cfrg-partially-blind-rsa/.

[7] Anonymous Tokens (AT) no date. https://github.com/google/anonymous-tokens.

[8] Apple. 2021. iCloud Private Relay Overview. https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf.

[9] arkworks no date. https://github.com/arkworks-rs.

[10] Niko Bari and Birgit Pfitzmann. 1997. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *EUROCRYPT'97 (LNCS, Vol. 1233)*, Walter Fumy (Ed.). Springer, Heidelberg, Konstanz, Germany, 480–494.

[11] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. 2002. The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme. In *FC 2001 (LNCS, Vol. 2339)*, Paul F. Syverson (Ed.). Springer, Heidelberg, Berlin, Heidelberg, 319–338.

[12] Mihir Bellare and Phillip Rogaway. 1996. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In *EUROCRYPT'96 (LNCS, Vol. 1070)*, Ueli M. Maurer (Ed.). Springer, Heidelberg, Saragossa, Spain, 399–416.

[13] Fabrice Benhamouda, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. 2022. Publicly verifiable anonymous tokens with private metadata bit. Cryptology ePrint Archive, Paper 2022/004. https://eprint.iacr.org/2022/004 https://eprint.iacr.org/2022/004.

[14] Dan Boneh, Sergey Gorbunov, Riad Wahby, Hoeteck Wee, Christopher Woor, and Zhenfei Zhang. 2023. BLS Signatures. https://datatracker.ietf.org/doc/draft-irtf-cfrg-bls-signature/.

[15] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *ASIACRYPT 2001 (LNCS, Vol. 2248)*, Colin Boyd (Ed.). Springer, Heidelberg, Gold Coast, Australia, 514–532.

[16] Nikita Borisov. no date. Metadata Forgery in Partially-Blind RSA Signatures. Personal Communication.

[17] Carlos Cardenas. no date. Cloud Security: The Challenges with Key Management in the Cloud (and everywhere else). https://www.tritondatacenter.com/blog/cloud-security-the-challenges-with-key-management-in-the-cloud-and-everywhere-else.

[18] Melissa Chase, F. Betül Durak, and Serge Vaudenay. 2023. Anonymous Tokens with Stronger Metadata Bit Hiding from Algebraic MACs. In *Advances in Cryptology – CRYPTO 2023*. Springer Nature Switzerland, Santa Barbara, USA, 418–449.

[19] David Chaum. 1982. Blind Signatures for Untraceable Payments. In *CRYPTO'82*, David Chaum, Ronald L. Rivest, and Alan T. Sherman (Eds.). Plenum Press, New York, USA, Santa Barbara, USA, 199–203.

[20] David Chaum. 1983. Blind Signature System. In *CRYPTO'83*, David Chaum (Ed.). Plenum Press, New York, USA, Santa Barbara, USA, 153.

[21] CIRCL (Cloudflare Interoperable, Reusable Cryptographic Library) no date. https://github.com/cloudflare/circl.

[22] Jean-Sébastien Coron and David Naccache. 2000. Security Analysis of the Gennaro-Halevi-Rabin Signature Scheme. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding (Lecture Notes in Computer Science, Vol. 1807)*. Springer, 91–101. https://doi.org/10.1007/3-540-45539-6_7

[23] Geoffroy Couteau, Thomas Peters, and David Pointcheval. 2017. Removing the Strong RSA Assumption from Arguments over the Integers. In *EUROCRYPT 2017, Part II (LNCS, Vol. 10211)*, Jean-Sébastien Coron and Jesper Buus Nielsen (Eds.). Springer, Heidelberg, Paris, France, 321–350.

[24] Cryptographic Module Validation Program: Modules In Process List no date. https://csrc.nist.gov/projects/cryptographic-module-validation-program/modules-in-process/modules-in-process-list. Accessed on 08.08.2024.

[25] Ivan Damgård and Eiichiro Fujisaki. 2002. A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. In *ASIACRYPT 2002 (LNCS, Vol. 2501)*, Yuliang Zheng (Ed.). Springer, Heidelberg, Queenstown, New Zealand, 125–142.

[26] Alex Davidson, Armando Faz-Hernandez, Nick Sullivan, and Christopher Wood. 2024. Oblivious Pseudorandom Functions (OPRFs) Using Prime-Order Groups. https://datatracker.ietf.org/doc/rfc9497/.

[27] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. 2018. Privacy Pass: Bypassing Internet Challenges Anonymously. *PoPETs* 2018, 3 (July 2018), 164–180. https://doi.org/10.1515/popets-2018-0026

[28] Randall Degges. no date. The Hardest Thing About Data Encryption. https://developer.okta.com/blog/2019/07/25/the-hardest-thing-about-data-encryption.

[29] Frank Denis, Frederic Jacobs, and Christopher Wood. 2023. RSA Blind Signatures. https://datatracker.ietf.org/doc/draft-irtf-cfrg-rsa-blind-signatures/.

[30] Sam Dutton. no date. Getting started with Trust Tokens. https://web.dev/trust-tokens/

[31] Eiichiro Fujisaki and Tatsuaki Okamoto. 1997. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In *CRYPTO'97 (LNCS, Vol. 1294)*, Burton S. Kaliski Jr. (Ed.). Springer, Heidelberg, Santa Barbara, USA, 16–30. https://doi.org/10.1007/BFb0052225

[32] Rosario Gennaro, Shai Halevi, and Tal Rabin. 1999. Secure Hash-and-Sign Signatures Without the Random Oracle. In *Advances in Cryptology — EUROCRYPT '99*, Jacques Stern (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 123–139.

[33] Google. no date. BoringSSL. https://github.com/google/boringssl.

[34] Google Cloud pricing no date. https://cloud.google.com/pricing.

[35] Scott Hendrickson and Christopher Wood. 2023. Privacy Pass Issuance Protocols with Public Metadata. https://datatracker.ietf.org/doc/draft-ietf-privacypass-public-metadata-issuance/.

[36] Scott Hendrickson and Christopher A. Wood. 2024. *Privacy Pass Issuance Protocols with Public Metadata*. Internet-Draft draft-ietf-privacypass-public-metadata-issuance-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-privacypass-public-metadata-issuance/

[37] Sharon Huang, Subodh Iyengar, Sundar Jeyaraman, Shiv Kushwah, Chen-Kuei Lee, Zutian Luo, Payman Mohassel, Ananth Raghunathan, Shaahid Shaikh, Yen-Chieh Sung, et al. 2021. Dit: De-identified authenticated telemetry at scale.

[38] Julia Kastner, Julian Loss, and Jiayu Xu. 2022. On pairing-free blind signature schemes in the algebraic group model. In *IACR International Conference on Public-Key Cryptography*. Springer, 468–497.

[39] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. 2020. Anonymous Tokens with Private Metadata Bit. In *CRYPTO 2020, Part I (LNCS, Vol. 12170)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Heidelberg, Santa Barbara, USA, 308–336.

[40] Anna Lysyanskaya. 2023. Security Analysis of RSA-BSSA. In *IACR International Conference on Public-Key Cryptography*. Springer-Verlag, Atlanta, USA, 251–280. https://doi.org/10.1007/978-3-031-31368-4_10

[41] Pieter Moree. 2014. Integers without large prime factors: from Ramanujan to de Bruijn. *Integers* 14 (2014).

[42] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. 2016. PKCS #1: RSA Cryptography Specifications Version 2.2. https://datatracker.ietf.org/doc/html/rfc8017.

[43] Network Security Services no date. https://github.com/nss-dev/nss.

[44] Google One. no date. VPN by Google One, explained. https://one.google.com/about/vpn/howitworks.

[45] OpenSSL no date. https://github.com/openssl/openssl.

[46] Michele Orrù, Stefano Tessaro, Greg Zaverucha, and Chenzhi Zhu. 2024. Oblivious issuance of proofs. IACR Crypto 2024, Springer-Verlag.

[47] Pairing on the BLS12-381 Elliptic Curve no date. https://github.com/nccgroup/pairing-bls12381.

[48] David Pointcheval and Jacques Stern. 1996. Provably Secure Blind Signature Schemes. In *ASIACRYPT'96 (LNCS, Vol. 1163)*, Kwangjo Kim and Tsutomu Matsumoto (Eds.). Springer, Heidelberg, Kyongju, Korea, 252–265. https://doi.org/10.1007/BFb0034852

[49] POPRF Implementation no date. https://github.com/facebook/voprf.

[50] Luke Probasco. no date. Key Management: The Hardest Part of Encryption. https://info.townsendsecurity.com/bid/74336/key-management-the-hardest-part-of-encryption.

[51] RELIC toolkit no date. https://github.com/relic-toolkit/relic.

[52] Dominique Schröder and Dominique Unruh. 2012. Security of Blind Signatures Revisited. In *PKC 2012 (LNCS, Vol. 7293)*, Marc Fischlin, Johannes Buchmann, and Mark Manulis (Eds.). Springer, Heidelberg, Darmstadt, Germany, 662–679.

[53] Atle Selberg. 1949. An elementary proof of the prime-number theorem. *Annals of Mathematics* 50, 2 (1949), 305–313.

[54] Victor Shoup. 2000. Practical Threshold Signatures. In *Advances in Cryptology — EUROCRYPT 2000*. Springer Berlin Heidelberg, Berlin, Heidelberg, 207–220.

[55] Victor Shoup. 2009. *A computational introduction to number theory and algebra*. Cambridge university press, Cambridge.

[56] Tjerand Silde and Martin Strand. 2022. Anonymous Tokens with Public Metadata and Applications to Private Contact Tracing. In *Financial Cryptography and Data Security*. Springer International Publishing, Cham, 179–199.

[57] Stefano Tessaro and Chenzhi Zhu. 2022. Short pairing-free blind signatures with exponential security. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 782–811.
[58] The Bouncy Castle Crypto Package For Java no date. https://github.com/bcgit/bc-java.
[59] The MIRACL Core Cryptographic Library no date. https://github.com/miracl/core.
[60] Nirvan Tyagi, Sofia Celi, Thomas Ristenpart, Nick Sullivan, Stefano Tessaro, and Christopher A. Wood. 2022. A Fast and Simple Partially Oblivious PRF, with Applications. In *Advances in Cryptology – EUROCRYPT 2022*. Springer International Publishing, Trondheim, Norway, 674–705.
[61] John Wilander. 2021. Introducing Private Click Measurement, PCM. https://webkit.org/blog/11529/introducing-private-click-measurement-pcm/
[62] wolfSSL Embedded SSL/TLS Library no date. https://github.com/wolfSSL/wolfssl.
[63] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. 2003. Efficient Verifiably Encrypted Signature and Partially Blind Signature from Bilinear Pairings. In *INDOCRYPT 2003 (LNCS, Vol. 2904)*, Thomas Johansson and Subhamoy Maitra (Eds.). Springer, Heidelberg, New Delhi, India, 191–204.

A Discussion about Strong RSA Modulus

Generating Strong RSA Modulus. For our new protocol, we need to modify the Gen algorithm to always output strong RSA modulus $N = pq$ where p and q are safe primes. The straightforward approach is to use the same Gen algorithm of generating random primes and simply check whether the chosen primes are safe. For any prime p, one can simply using primality testing algorithms to check whether $(p-1)/2$ is prime to see if p is a safe prime. As safe primes are more scarce than primes, this algorithm will be slower. However, we can estimate the asymptotic additional time needed to generate safe primes compared to normal primes. By the prime number theorem [53], we know that the density of prime numbers is $O(1/\kappa)$ for numbers at most 2^κ. For safe primes, we can rely on one of several conjectures e.g. Dickson's conjecture or the twin prime conjecture ([55]) that state the density of safe primes is $O(1/\kappa^2)$. Therefore, we could expect that the Gen algorithm finds a safe prime after generating $O(\kappa)$ random primes.

Next, we compare the Standard RSA Assumption to the one restricted to only Strong RSA Modulus.

Proof of Theorem 3.4. First, we will assume there is an adversary $\mathcal{A}$ that breaks the strong modulus version of the assumption with probability ϵ and running time t. We show that $\mathcal{A}$ will also break the standard RSA game with probability $O(\epsilon/\kappa^2)$ and running time t to prove the statement. Recall that κ is the length of the prime factors of the RSA modulus. To prove this, we simply analyze the probability that a randomly generated RSA modulus is a strong RSA modulus. Assuming Dickson's conjecture [55], the probability that a randomly generated prime number of length κ is also a safe prime is $\Theta(1/\kappa)$. For a RSA modulus to be strong, both prime factors must be safe. Therefore, this occurs with probability $\Theta(1/\kappa^2)$. Conditioned on the RSA modulus being strong, the adversary $\mathcal{A}$ wins the standard RSA game with probability ϵ. Therefore, $\mathcal{A}$ wins the game with probability at least $O(\epsilon/\kappa^2)$. □

To our knowledge, there is no straightforward relation in the opposite direction. We leave it as an open question to determine the relation in this direction.

B Equivalence of Unforgeability Definitions

In this section, we consider the alternative definition of unforgeability with public metadata that was introduced in [56]. At a high level, the difference is that this alternative definition does not limit the total number of queries explicitly. Instead, the adversary is limited to at most ℓ oracle signing queries for each choice of public metadata D. Moreover, to win the game, adversary needs to output $\ell+1$ signature message pairs under a fixed choice of metadata. We denote this definition as Alternative Strong One-More Unforgeability with Public Metadata, $\mathrm{G}^{\mathsf{ASOMUF}}_{\mathsf{Tok},\mathcal{A},\ell}(\lambda)$.

Theorem B.1. *A token scheme* Tok *satisfies alternative strong one-more unforgeability with public metadata if and only if the token scheme* Tok *also satisfies (Def. 2.2). The reductions only lose up to $O(t+\ell)$ additive factors in the adversary's running time and number of oracle signing queries.*

Proof. Assume there exists an adversary $\mathcal{A}$ that wins the game $\mathrm{G}^{\mathsf{ASOMUF}}$. The same adversary also wins the game $\mathrm{G}^{\mathsf{SOMUF}}$ as the maximum number of oracle queries sent by the adversary is at most its running time. In the other direction, assume that $\mathcal{A}$ wins the game $\mathrm{G}^{\mathsf{SOMUF}}$. If the output of $\mathcal{A}$ consists of $\ell+1$ tuples, then it also wins $\mathrm{G}^{\mathsf{ASOMUF}}$. Otherwise, we can afford to use additional queries to the signing oracle in $\mathrm{G}^{\mathsf{ASOMUF}}$. Suppose that $\mathcal{A}$ outputs cnt_D+1 tuples using at most cnt_D signing oracle queries for public metadata D. Then, we can win $\mathrm{G}^{\mathsf{ASOMUF}}$ by the adversary making an additional $\ell-\mathsf{cnt}_D-1$ signing oracle queries for different messages. Therefore, Tok is not $(\epsilon, t+O(\ell), 2\ell)$-alternative strong one-more unforgeable as we bound the additional running time by $O(\ell)$ and the additional number of signing oracle queries by at most ℓ. □

Winning Conditions for RSA Inversion Games. We note the same equivalence can be shown to exist for the RSA assumption games. In our games, the adversary's condition for winning is outputting the tuple $(X_i, Y_i)_{i\in[z]}$ where $z > \mathsf{cnt}_e$ where cnt_e oracle queries are done. For some prior works, the adversary's winning condition is outputting a pair (X, Y) for e that is different from any query to O^{RSA}. We show the two are essentially equivalent. Since there are only cnt_e queries to O^{RSA} for e, the tuple $(X_i, Y_i)_{i\in[z]}$ must contain at least one pair (X, Y) that was never queried. If an adversary is able to find a single pair (X, Y) that was never queried to O^{RSA}, the adversary can pad the results of the cnt_e oracle queries for e to obtain a tuple of $z > \mathsf{cnt}_e$ pairs.

C Equivalence of RSA and Multi-Exponent RSA Assumption

In this section, we present the proof of Theorem F.1 where we show that the RSA assumption (Def. 3.3) and the multi-exponent RSA assumption (Def. 4.1) are equivalent up to a multiplicative ℓ factor when the number of exponents is ℓ in the latter assumption.

Proof of Theorem 4.2. Towards a contradiction, suppose that there exists an adversary $\mathcal{A}'$ that wins the multi-exponent RSA game with probability ϵ' with running time t' when given ℓ challenge exponents. We build an adversary $\mathcal{A}$ for the RSA game that wins with probability $\epsilon = \epsilon'/\ell$ with running time $t = t' + O(\ell)$. Recall that $\mathcal{A}$ is given an exponent e from $[3, e_{\max}]$ and random target X and must produce Y satisfying $Y^e = X \bmod N$. To do this, $\mathcal{A}$ will choose ℓ challenge exponents $e_1, \ldots, e_\ell$ as follows. $\mathcal{A}$ first picks a uniformly random number from $z \in [\ell]$. Then, $\mathcal{A}$ will set $e_z = e$. For the remaining $i \in [\ell] \setminus \{z\}$, $\mathcal{A}$ sets e_i to a random odd number from the set $[3, e_{\max}]$. As we assume N is a

Game $\mathrm{G}^{\mathrm{SRSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$:
$(N,(p,q)) \leftarrow_R \mathsf{Gen}(1^\lambda)$
$X \leftarrow_R \mathbb{Z}^*_N$
$(e, Y) \leftarrow \mathcal{A}(\mathsf{crs}, N, X)$
Return 1 if and only if $e \geq 3$ and $Y^e = X \bmod N$.

Figure 12: Strong RSA Game.

strong RSA modulus and the product of two κ-bit primes, we know that $\phi(N) = 4p'q'$ for some primes p' and q' of length at least $\kappa - 1$. Therefore, odd numbers from $[3, 2^{\kappa-2}]$ will be co-prime with $\phi(N)$ and, thus, valid exponents. As each e_i are chosen such that $e_i \leq e_{\mathbf{max}} \leq 2^{\kappa-2}$, we know that each e_i is a valid exponent. Finally, $\mathcal{A}$ passes the modulus N, the ℓ exponents as well as the random target X to $\mathcal{A}'(N, e_1, \ldots, e_\ell, X)$. First, we note that the distribution of inputs seen by $\mathcal{A}'$ is identical to the multi-exponent RSA game (see Figure 5). $\mathcal{A}'$ successfully forges and, thus, returns i as well as a valid decryption of X under e_i with probability ϵ'. In other words, $\mathcal{A}'$ outputs Y satisfying $Y = X^{1/e_i}$. Therefore, $\mathcal{A}$ can output Y to win the RSA game as long as $i = z$ with probability ϵ'/ℓ. If $\mathcal{A}'$ runs in time t', then $\mathcal{A}$ runs in time $t' + O(\ell)$ with $O(\ell)$ time to generate random exponents. □

D Connections to Strong RSA Assumption

In this section, we explore two variants of one-more strong RSA assumptions where we provide adversaries more freedom compared to the assumption we use to prove security of our protocols (Def. 4.3). In both cases, there exists a polynomial time adversary that can break the assumption. For completeness, we also present the game for the strong RSA assumption in Figure 12.

One-More RSA with Chosen Exponents. As our first definition, we will consider the game in Figure 6 with the following modifications. The decryption oracle, O^{RSA}, will now allow the adversary to submit arbitrary exponents for decryption. As a technical detail, we do require queried exponents to be co-prime to $\phi(N)$ to enable decryption. In the case that the input exponent is not co-prime to $\phi(N)$, O^{RSA} will output $\perp$. Note, we will still require the adversary to output exponents from O^{exp} to win the game. Clearly, this is a stronger assumption compared to the chosen-target, restricted-exponent assumption (Definition 4.3) as the adversary has significantly more freedom when choosing decryption oracle queries. We formally denote this modified game as the chosen-target, chosen-exponent RSA inversion game $\mathrm{G}^{\mathsf{CT\text{-}CE\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$. We use the notion *chosen-exponent* to denote that the adversary is now free to pick any exponents to submit to the decryption oracle O^{RSA}. The full game is presented in Figure 13.

Definition D.1 ((ϵ, t, ℓ)-Chosen-Target, Chosen-Exponent RSA Inversion Assumption). Let λ be the security parameter and consider $\mathrm{G}^{\mathsf{CT\text{-}CE\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 13. The $(\epsilon, t, \ell, \ell')$-chosen-target, chosen-exponent RSA inversion assumption is true for Gen if, for any adversary $\mathcal{A}$ that runs in time t, makes at most ℓ decryption queries and ℓ' exponent queries, then $\Pr[\mathrm{G}^{\mathsf{CT\text{-}CE\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon$.

While this seems like a reasonable assumption, we can show that this already provides far too much power to an adversary.

Theorem D.2. *There exists a polynomial time adversary $\mathcal{A}$ that wins the chosen-target, chosen-exponent RSA inversion game.*

Proof. We present a simple adversary $\mathcal{A}$ as follows. First, $\mathcal{A}$ gets a target $X \leftarrow O^{\mathcal{X}}()$. Next, the $\mathcal{A}$ gets $e \leftarrow O^{\mathrm{exp}}()$ and picks a random number e' that are both co-prime to $\phi(N)$. Without loss of generality, let $e' > e$. The adversary calls the decryption oracle to receive $Y \leftarrow O^{\mathrm{RSA}}(X^{e'}, e \cdot e')$. Then, $Y = (X^{e'})^{1/(e \cdot e')} = X^{1/e}$. Finally, $\mathcal{A}$ outputs $e, (X, Y)$ and wins the game as the adversary never made a single decryption oracle call for e. □

Interestingly, the ability for the adversary to choose arbitrary exponents to send to the decryption oracle enables far too much adversarial power. The same reasoning was used in prior RSA assumptions where if the adversary is able to pick arbitrary messages to decrypt, then the adversary can easily break prior RSA assumptions. While this does not happen for exponents in the standard assumptions without decryption oracles, it ends up being the case once you provide decryption oracles to the adversary.

One-More RSA with Arbitrary Exponents. In our second attempt, we no longer provide a challenge exponent oracle, O^{exp}, to the adversary. Instead, the adversary is free to choose any exponent $e \geq 3$ and produce a decryption with respect to e. We denote this game as the chosen-target, arbitrary-exponent RSA inversion game. Clearly the adversary can win this game as well, as it won the previous game where it had less adversarial power.

E RSA Signatures with Public Metadata: Proof of Unforgeability

In this section, we prove the concrete security of our RSA signature with public metadata from the RSA assumption. We show that the concrete security bounds are similar to those proven by [12] for standard RSA signatures with FDH message encodings except for a multiplicative factor loss in number of hashing oracle calls to H_{MD}. At a high level, our security proof will construct an adversary $\mathcal{A}$ for the multi-exponent RSA game given exponents from $[3, 2^{\kappa-2}]$. Afterwards, we can apply Theorem 4.2 to obtain security based on the standard RSA assumption. We start with the first step reducing security to the multi-exponent RSA assumption:

Lemma E.1. *Suppose that $(H_{\mathcal{M}}, \mathsf{Verify}_{\mathcal{M}})$ correspond to full domain hash (FDH) message encoding. Assuming the $(\epsilon, t, q_{\mathsf{MD}})$-multi-exponent RSA assumption (Definition 4.1) with exponents in $[3, 2^{\kappa-2}]$ and the random oracle model, then $\mathsf{RSA}_{\mathsf{MD}}$ is $(\epsilon_F, t_F, \ell_F)$-strong one-more unforgeable (Definition 2.2) where*

- $t_F = t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$
- $\epsilon_F = q_{\mathcal{M}} \cdot \epsilon$

and the adversary $\mathcal{A}$ runs in time at most t_F and makes at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle and hash functions $H_{\mathcal{M}}$ and H_{MD} respectively.

Proof. Let $\mathcal{A}_F$ be an adversary that can break the $(\epsilon_F, t_F, \ell_F)$-SOMUF of $\mathsf{RSA}_{\mathsf{MD}}$ with at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle, $H_{\mathcal{M}}$ and H_{MD}. That is, $\mathcal{A}_F$ can forge a signature with success probability of ϵ_F with running time of t_F. To complete the proof, we show that one can use $\mathcal{A}_F$ to construct an adversary $\mathcal{A}$ that can break the multi-exponent RSA game with exponents

Game $\mathrm{G}^{\text{CT-CE-RSA}}_{\text{Gen},\mathcal{A}}(\lambda)$:	Oracle $\mathcal{O}^{\text{RSA}}(X, e)$:	Oracle $\mathcal{O}^{X}()$:	Oracle $\mathcal{O}^{\text{exp}}()$:
$(N, \mathcal{D}_N, (p, q)) \leftarrow_R \text{Gen}(1^\lambda)$ $\mathcal{S}_X \leftarrow \emptyset, \mathcal{S}_{\text{exp}} \leftarrow \emptyset$ $\phi(N) \leftarrow (p-1)(q-1)$ $\text{cnt}_e \leftarrow 0, \forall e \in \mathbb{Z}^*_{\phi(N)}$ $e, (X_i, Y_i)_{i\in[x]} \leftarrow \mathcal{A}^{\mathcal{O}^{\text{RSA}}, \mathcal{O}^X, \mathcal{O}^e}(N)$ **Return** 1 if and only if all the following hold: - $\text{cnt}_e < x$ - $e \in \mathcal{S}_{\text{exp}}$ - $\forall i \neq j \in [x], X_i \neq X_j$ - $\forall i \in [x], X_i \in \mathcal{S}_X$ - $\forall i \in [x], Y_i^e = X_i \bmod N$	If $e \notin \mathbb{Z}^*_{\phi(N)}$: **Return** $\perp$ $\text{cnt}_e \leftarrow \text{cnt}_e + 1$ $d \leftarrow e^{-1} \bmod \phi(N)$ $Y \leftarrow X^d \bmod N$ **Return** Y	$X \leftarrow_R \mathbb{Z}^*_N$ $\mathcal{S}_X \leftarrow \mathcal{S}_X \cup \{X\}$ **Return** X	$e \leftarrow_R \mathcal{D}_N$ $\mathcal{S}_{\text{exp}} \leftarrow \mathcal{S}_{\text{exp}} \cup \{e\}$ **Return** e

Figure 13: Chosen-Target, Chosen-Exponent RSA Inversion Game. All differences with the chosen-target, restricted-exponent RSA inversion game, $\mathrm{G}^{\text{CT-RE-RSA}}_{\text{Gen},\mathcal{A}}(\lambda)$, are highlighted in blue.

from $[3, 2^{\kappa-2}]$ using $O(q_{\mathcal{M}} + q_{\text{MD}} + \ell_F)$ additional running time and wins the game with probability at least $\epsilon_F/q_{\mathcal{M}}$.

For any adversary $\mathcal{A}$ trying to solve the multi-exponent RSA problem, $\mathcal{A}$ receives as input the challenge RSA exponents $(e_1, \ldots, e_{q_{\text{MD}}})$, strong RSA modulus N and a challenge target X that is chosen uniformly at random $\mathbb{Z}^*_N$. Recall that the goal is to pick any $i \in [q_{\text{MD}}]$ and output the e_i-th root of X modulo N. In other words, compute Y such that $Y^{e_i} = X \bmod N$.

$\mathcal{A}$ will execute the adversary $\mathcal{A}_F$ that outputs a forgery of the form $D, (S_i, M_i)_{i\in[x]}$. Without loss of generality, we will assume the following. First, $x \leq \ell_F + 1$. As $\mathcal{A}_F$ makes at most ℓ_F signing oracle queries, $\mathcal{A}$ will only need to produce at most $\ell_F + 1$ valid signatures to win the strong one-more unforgeability game. Secondly, we will assume that $\mathcal{A}_F$ will have queried $H_{\mathcal{M}}(M_i)$ on all output messages. Again, this is without loss of generality as it will only increase the running time of $\mathcal{A}_F$ by at most $\ell_F + 1$ hash function queries. Furthermore, we will suppose that $\mathcal{A}_F$ will have queried both $H_{\mathcal{M}}(M)$ and $H_{\text{MD}}(D)$ before submitting a signing query for M and D. Again, this is without loss of generality, as it only increases the running time of $\mathcal{A}_F$ by at most $2\ell_F$ hash queries.

During the execution of $\mathcal{A}_F$, $\mathcal{A}$ must successfully simulate queries for signing as well as the message encoding hash function $H_{\mathcal{M}}$ and the public metadata hash function H_{MD}. Before doing so, $\mathcal{A}$ keeps track of a count of the number of unique inputs to the message oracle $H_{\mathcal{M}}$. We know that at most $q_{\mathcal{M}}$ queries are made to $H_{\mathcal{M}}$. $\mathcal{A}$ will pick a random integer $z \leftarrow_R [q_{\mathcal{M}}]$ and will embed the challenge X as the output of the z-th query to $H_{\mathcal{M}}$. To keep track, $\mathcal{A}$ keeps counter $\text{cnt}_{\mathcal{M}}$ to count the number of unique inputs to $H_{\mathcal{M}}$ where $\text{cnt}_{\mathcal{M}}$ is initialized to zero. Similarly, $\mathcal{A}$ uses cnt_{MD} to count the number of unique inputs to H_{MD} with cnt_{MD} also initialized to zero. Finally, $\mathcal{A}$ keeps a map of inputs to important information for the hash function H_{MD} and $H_{\mathcal{M}}$ denoted by $\mathbf{M}_{\text{MD}}$ and $\mathbf{M}_{\mathcal{M}}$. $\mathcal{A}$ will simulate each of these queries as follows.

For any query to $H_{\text{MD}}(D)$:

(1) If $\mathbf{M}_{\text{MD}}[D]$ is set, return $\mathbf{M}_{\text{MD}}[D]$.
(2) Increment $\text{cnt}_{\text{MD}} \leftarrow \text{cnt}_{\text{MD}} + 1$ and set $\mathbf{M}_{\text{MD}}[D] \leftarrow e_{\text{cnt}_{\text{MD}}}$.
(3) Return $\mathbf{M}_{\text{MD}}[D]$.

For any query to $H_{\mathcal{M}}(M, D)$:

(1) Compute $e_{\text{MD}} \leftarrow H_{\text{MD}}(D)$.
(2) If $\mathbf{M}_{\mathcal{M}}[M, D]$ is set, return first value of tuple $\mathbf{M}_{\mathcal{M}}[M, D]$.
(3) Increment $\text{cnt}_{\mathcal{M}} \leftarrow \text{cnt}_{\mathcal{M}} + 1$.
(4) If $\text{cnt}_{\mathcal{M}} = z$:
  (a) Set $\mathbf{M}_{\mathcal{M}}[M, D] \leftarrow (X, \perp)$.
(5) Otherwise when $\text{cnt}_{\mathcal{M}} \neq z$:
  (a) Generate random $A \leftarrow_R \mathbb{Z}^*_N$.
  (b) Compute $B = A^{e_{\text{MD}}} \bmod N$.
  (c) Set $\mathbf{M}_{\mathcal{M}}[M, D] \leftarrow (B, A)$.
(6) Return first value of tuple $\mathbf{M}_{\mathcal{M}}[M, D]$.

For any query to $\mathcal{O}^{\text{Sign}}(M, D)$:

(1) Retrieve $(B, A) \leftarrow \mathbf{M}_{\mathcal{M}}[M, D]$.
(2) If $A = \perp$, abort. Else, return A.

Finally, we will suppose that $\mathcal{A}$ does not abort and $\mathcal{A}_F$ outputs some forgery $D, (S_i, M_i)_{i\in[x]}$ while making strictly less than x signing queries for metadata D. If $H_{\mathcal{M}}(M_i \,||\, D) = X$ for any $i \in [x]$, then $H_{\mathcal{M}}(M_i \,||\, D) = X = S_i^{H_{\text{MD}}(D)}$ assuming that $\mathcal{A}_F$ is successful at forging. Re-writing the above, we get the following equality $S_i = X^{1/H_{\text{MD}}(D)} = X^{1/e_j}$ for some $j \in [q_{\text{MD}}]$ and e_j is the j-th challenge exponent. In other words, this is a successful decryption of the ciphertext X that would win the multi-exponent RSA game. Therefore, $\mathcal{A}$ will simply return (j, S_i) that would win the multi-exponent RSA game.

Adversarial Advantage. First, we show that the simulated view and real view of $\mathcal{A}_F$ are identical. In the real game, $\mathcal{A}_F$ receives random exponents from $[3, 2^{\kappa-2}]$. In the simulated game, $\mathcal{A}_F$ receives random challenge exponents $e_1, \ldots, e_{q_{\text{MD}}}$ that are also random exponents from $[3, 2^{\kappa-2}]$. The view from $H_{\mathcal{M}}$ and the signing oracle are also identical as long as $\mathcal{A}$ does not abort. To upper bound the aborting probability, we first note that $\mathcal{A}_F$ must return at least one message M_i such that (M_i, D) was never sent to the signing oracle. Assuming that $\mathcal{A}$ successfully executed $\mathcal{A}_F$ without aborting, the probability that public metadata D and message M_i are output by $\mathcal{A}_F$ where $H_{\mathcal{M}}(M_i \,||\, D) = X$ such that (M_i, D) was not a signing oracle query by $\mathcal{A}_F$ is at least $1/q_{\mathcal{M}}$. This is because $\mathcal{A}_F$ makes at most $q_{\mathcal{M}}$ queries to $H_{\mathcal{M}}$ and must output at least one message. In this case, it is clear that $\mathcal{A}$ will not abort when executing $\mathcal{A}_F$. Finally, $\mathcal{A}$ produces a forgery with probability at most ϵ_F meaning that $\mathcal{A}$ wins the RSA game with probability at least $\epsilon_F/q_{\mathcal{M}}$.

Adversarial Running Time. Suppose that $\mathcal{A}_F$ runs in time t_F. By our assumptions, we note that we increase the running time of $\mathcal{A}_F$ by at most $O(\ell_F)$ hash queries. For each signing query, $\mathcal{A}$ must perform a single exponentiation. For each query to $H_{\mathcal{M}}$ and H_{MD},

$\mathcal{A}$ requires $O(1)$ time to simulate each answer correctly. Therefore, $\mathcal{A}$ requires $t_F + O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$ time. □

Using this lemma, we can apply Theorem 4.2 to reduce security to standard RSA and prove our main theorem about unforgeability.

Proof of Theorem 5.3. We apply Lemma E.1 to reduce security to the $(\epsilon_F/q_{\mathcal{M}}, t_F + O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F), q_{\mathsf{MD}})$-multi-exponent RSA assumption with exponents from $[3, 2^{\kappa-2}]$. By applying Theorem 4.2, we obtain that this is equivalent to the $(\epsilon_F/(q_{\mathsf{MD}} \cdot q_{\mathcal{M}}), t_F + O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F))$-RSA assumption with exponents from $[3, 2^{\kappa-2}]$. □

Encoding of Message and Public Metadata. We note that our proof critically uses the fact that we pass both the message and public metadata into the FDH encoding algorithm. When $\mathcal{A}$ programs the random oracle for $H_{\mathcal{M}}$, $\mathcal{A}$ exponentiates according to $H_{\mathsf{MD}}(D)$. This is only possible because D is also passed as an input to $H_{\mathcal{M}}$. If, instead, we only passed the message M into $H_{\mathcal{M}}$, then the best that $\mathcal{A}$ can do is simply aim to guess the public metadata that will be later sent with M to the signing oracle. As there is no requirement that $\mathcal{A}_F$ has even picked D yet, it is impossible for $\mathcal{A}$ to guess this correctly. Therefore, it is integral that our algorithm passes the public metadata D into $H_{\mathcal{M}}$ for security.

Extension to PSS Encoding. Finally, we note that our proof can be modified when using PSS message encoding. In particular, the random oracle $H_{\mathcal{M}}$ is currently programmed in a trivial way to output random elements that is indistinguishable from FDH encodings. Instead, we can follow the identical proof techniques of Bellare and Rogaway [12] to program $H_{\mathcal{M}}$ to follow the PSS encoding. The main difference is the following. In the FDH proof, the adversary had to pick one of the query to $H_{\mathcal{M}}$ to choose to embed the random input target. By using the structure of PSS encoding, the adversary can instead embed the random input target into every query to $H_{\mathcal{M}}$. This significantly improves the ability of the adversary to win the game. As a result, we can obtain identical concrete security when using PSS encoding as those proved in [12] except for a similar additive factor exponentially small in κ when simulating the public metadata hash function H_{MD}.

E.1 Non-Applicability of Multi-Exponent Attacks for RSA Signatures

In this section, we explain in detail why the multi-exponent attacks in Section 7 that leverage certain algebraic properties that appear in exponents are not applicable for our standard (non-blind) RSA signatures with public metadata. At a high level, the reasoning lies in one of the core difference between standard and blind RSA signatures. In standard (non-blind) signature schemes, the signer receives the input message M along with the public metadata D in plaintext. In contrast, for blind RSA signatures, the signer receives an element X in the RSA group that is a blinded version of the input message M (the public metadata D is still received in plaintext). This critical difference is leveraged by the multi-exponents attacks in Section 7 as we will show.

Taking a closer look at the attack, it requires find a sequence $D, D_1, \ldots, D_z$ of public metadata with the following properties. Let $e = H_{\mathsf{MD}}(D) = p_1 \cdots p_z$ be the product of (not necessarily distinct) primes such that $p_i \mid e_i$ where $e_i = H_{\mathsf{MD}}(D_i)$. Next, we can consider the first iteration of the loop of the attack. In particular, we set $x_0 = H_{\mathcal{M}}(M \,||\, D)$ for some input message M. In the next step, we need to obtain a valid signature of $x_0^{e_1/p_1}$ for the exponent $e_1 = H_{\mathsf{MD}}(D_1)$ associated with public metadata D_1. The output will essentially be x_1. The step is repeated to obtain x_i that is output of the blind signing protocol for element $x_{i-1}^{e_i/p_i}$ for all $i \in [z]$. The final output is x_z that is a forgery for message M and public metadata D. For the blind RSA signatures with public metadata protocol, we note that the signer receives a blinded version of the input message M that is equivalent to some element in the underlying RSA group. Therefore, it is perfectly valid to send $x_{i-1}^{e_i/p_i}$ as the input element for the blind signing protocol.

We can now attempt to do the same attack for the standard (non-blind) RSA signatures with public metadata. The key point is that the adversary can no longer submit the element $x_{i-1}^{e_i/p_i}$ directly as input to the signing protocol. If the adversary wished to perform blind signing for this element, it must find some input message M_i such that $H_{\mathcal{M}}(M_i \,||\, D_i) = x_{i-1}^{e_i/p_i}$. As we assume that $H_{\mathcal{M}}$ is a random oracle (that is, an ideal one-way function), it is computationally intractable for the adversary to find such a message M_i such that $H_{\mathcal{M}}(M_i \,||\, D_i) = x_{i-1}^{e_i/p_i}$. Therefore, this attack does not apply directly to our standard RSA signatures with public metadata.

As a result, we note that several features required for the blind RSA signatures with public metadata are not needed for our non-blind protocol. First, we do not require the exponent checks that re-sample salts for H_{MD} when the set of valid exponents satisfy the algebraic properties required for the attack above. Secondly, our underlying RSA assumptions can simply use random odd exponents (instead of only focusing on exponent sets without the algebraic property). Therefore, it is not surprising that we prove our standard (non-blind) protocol directly from the RSA assumption (Definition 3.3) while we require the algebraically-restricted exponent assumption (Definition 4.4) for our blind signatures.

F Modified RSA Signatures with Public Metadata with Better Concrete Security

In this section, we show that there is a slightly modified variant of the RSA signatures with public metadata protocol from Section 5 with better concrete security reductions. The resulting security bounds of this modified protocol identically match those obtained [12] for RSA signatures (without public metadata).

Modified Protocol. Our modified protocol makes two minor changes to the protocol presented in Section 5. First, the public key is augmented to contain both the strong RSA modulus N and standard RSA public exponent e. Additionally, we will now compute e_{MD} as $e \cdot H_{\mathsf{MD}}(D)$.

New Reduction to Multi-Exponent RSA Assumption. We consider a modified reduction from the RSA assumption to the multi-exponent RSA assumption from Theorem 4.2 under different exponent distributions. For the RSA assumption, we will assume that a fixed public exponent e as in practice. For the multi-exponent RSA assumption, we will use the exponent distribution $\mathcal{D}'_{N,e}$ defined as follows. An uniformly random odd number e' is chosen from $[3, 2^{\kappa-2}]$. The final output is $e \cdot e'$. We note that the output can be much larger than $\kappa - 2$ bits (as large as $2\kappa - 4$ bits). Nevertheless, it is not an issue as it will still be invertible modulo $\phi(N)$.

Theorem F.1. *If the (ϵ, t)-RSA assumption for strong modulus (Definition 3.3) is true with a fixed exponent e, then the $(\epsilon, t-O(\ell), \ell)$-multi-exponent RSA assumption (Definition 4.1) is true with exponents drawn from $\mathcal{D}'_{N,e}$ defined above.*

Proof. Suppose that there exists an adversary $\mathcal{A}'$ that wins the multi-exponent RSA game with probability ϵ' with running time t' when given ℓ challenge exponents. We build an adversary $\mathcal{A}$ for the RSA game that wins with probability ϵ' with the running time $t = t' + O(\ell)$. Recall that $\mathcal{A}$ is given an exponent e and random target X and must produce Y satisfying $Y^e = X \bmod N$. To do this, $\mathcal{A}$ will pick ℓ random odd numbers from the set $[1, 2^{\kappa-2}]$ that we denote as $z_1, \ldots, z_\ell$. We choose the ℓ challenge exponents as $e_i = e \cdot z_i$ for all $i \in [\ell]$. Note, each e_i are essentially valid exponents drawn from the distribution $\mathcal{D}'_{N,e}$. Finally, $\mathcal{A}$ passes the modulus N, the ℓ exponents as well as the random target X to $\mathcal{A}'(N, e_1, \ldots, e_\ell, X)$. We know that $\mathcal{A}'$ returns i and a valid decryption Y of X under e_i with probability ϵ'. Additionally, we know that $Y = X^{1/e_i} = X^{1/(e \cdot z_i)}$ in this case. $\mathcal{A}$ outputs $Y^{z_i} = X^{1/e}$ to win the RSA game with probability ϵ'. For the running time, $\mathcal{A}$ uses an additional $O(\ell)$ time to generate random exponents. □

Unforgeability. With the new reduction, we can essentially redo the proof of Theorem 5.3 using the above multi-exponent RSA assumption with the modified exponent distribution. The security bounds are identical to the ones proven in [12] for standard non-public metadata RSA signatures.

Lemma F.2. *Suppose that $(H_{\mathcal{M}}, \mathsf{Verify}_{\mathcal{M}})$ correspond to full domain hash (FDH) message encoding. Assuming the $(\epsilon, t, q_{\mathsf{MD}})$-multi-exponent RSA assumption (Definition 4.1) with exponents from $\mathcal{D}'_{N,e}$ and the random oracle model, then $\mathsf{RSA}_{\mathsf{MD}}$ is $(\epsilon_F, t_F, \ell_F)$-strong one-more unforgeable (Definition 2.2) where*

- $t_F = t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$
- $\epsilon_F = q_{\mathcal{M}} \cdot \epsilon$

and the adversary $\mathcal{A}$ runs in time at most t_F and makes at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle and hash functions $H_{\mathcal{M}}$ and H_{MD} respectively.

Proof. The proof follows identically as the proof of Lemma E.1. The only difference required is the observation that the public exponents of the modified algorithm are generated identically to $\mathcal{D}'_{N,e}$. □

Theorem F.3. *Suppose that $(H_{\mathcal{M}}, \mathsf{Verify}_{\mathcal{M}})$ correspond to full domain hash (FDH) message encoding. Assuming the (ϵ, t)-RSA assumption (Definition 3.3) with fixed public exponent e and the random oracle model, then $\mathsf{RSA}_{\mathsf{MD}}$ is $(\epsilon_F, t_F, \ell_F)$-strong one-more unforgeable (Definition 2.2) where*

- $t_F = t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$
- $\epsilon_F = q_{\mathcal{M}} \cdot \epsilon$

and the adversary $\mathcal{A}$ runs in time at most t_F and makes at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle and hash functions $H_{\mathcal{M}}$ and H_{MD} respectively.

Proof. First, we apply Lemma F.2 to reduce security to the $(\epsilon_F/q_{\mathcal{M}}, t_F + O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F), q_{\mathsf{MD}})$-multi-exponent RSA assumption with exponents $\mathcal{D}'_{N,e}$ for some public exponent e. By applying Theorem F.1, we obtain that this is equivalent to the $(\epsilon_F/q_{\mathcal{M}}, t_F + O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F))$-RSA assumption with fixed exponent e. □

G Potential Denial-of-Service Risks

The protocols in this paper require performing RSA operations with larger exponents than standard RSA. The usage of large exponents may incur additional computation for the signer in the protocols but note that a user cannot change the size of the exponent (which is fixed at setup) arbitrarily. In particular, one could consider denial-of-service (DoS) risks where users wish to pick bad exponents (i.e., metadata) that may incur additional computation by the signer. In general, these risks are mitigated by the fact that the signer only performs computation on valid metadata $D \in \mathbf{MD}$. If an attacker picks bad metadata $D \notin \mathbf{MD}$, the signer rejects and avoids additional computation. Therefore, an attacker is restricted to trying to pick the worse metadata (and corresponding exponent) that appears in the valid metadata set $\mathbf{MD}$. If $\mathbf{MD}$ is small, this severely restricts the freedom of the attacker to find bad inputs to cause larger signer computation. Furthermore, if this is a concern, one can also re-sample the salt corresponding to H_{MD} and check that no exponents corresponding to valid metadata incurs large computation.

H Supplementary Material for Analyzing Multi-Exponent Attacks

H.1 Asymptotic Analysis of Exponent Check

We can also prove an asymptotic bound on the above probability using analytical upper bounds on Dickman's rho function [41]. In general, these upper bounds are much looser than the actual values of Dickman's rho function. Therefore, one should use the Dickman's rho function directly instead of the asymptotic upper bound as we will see later.

Theorem H.1. *Consider security parameter λ and public metadata set $\mathbf{MD}$ of size $|\mathbf{MD}| = \mathrm{poly}(\lambda)$. Suppose that H_{MD} outputs γ-bit odd integers where $\gamma = \kappa - 2$ and $\gamma \geq 32$. Let E be the event defined in Theorem 7.1. Then, for any randomly chosen salt, $\Pr[E] \leq O(2^{-\sqrt{\kappa}})$.*

Proof. We use the asymptotic upper bound from [41] stating that $\rho(u) = u^{-u+o(u)}$. We set $x = 2^{-\sqrt{\kappa}}$ in Theorem 7.1:

$$\begin{aligned}\Pr[E] &\leq \frac{|\mathbf{MD}|^2}{2^{\sqrt{\kappa}}} + 2|\mathbf{MD}| \cdot \rho(\gamma/\sqrt{\kappa}) \\ &\leq \frac{\mathrm{poly}(\lambda)}{2^{\sqrt{\kappa}}} + \frac{\mathrm{poly}(\lambda)}{(\gamma/\sqrt{\kappa})^{\gamma/\sqrt{\kappa}}}\end{aligned}$$

since $|\mathbf{MD}| = \mathrm{poly}(\lambda)$. Next, we plug in $\gamma = \kappa - 2$ and use the fact that $\gamma \geq 32$ to get

$$\begin{aligned}\Pr[E] &\leq \frac{\mathrm{poly}(\lambda)}{2^{\sqrt{\kappa}}} + \frac{\mathrm{poly}(\lambda)}{(\sqrt{\kappa}/2)^{\sqrt{\kappa}/2}} \\ &\leq \frac{\mathrm{poly}(\lambda)}{2^{\sqrt{\kappa}}} + \frac{\mathrm{poly}(\lambda)}{2^{\sqrt{\kappa}/2 \cdot \log(\sqrt{\kappa})-1}} \\ &\leq \frac{\mathrm{poly}(\lambda)}{2^{\sqrt{\kappa}}} + \frac{\mathrm{poly}(\lambda)}{2^{\sqrt{\kappa}}} \\ &= O\left(2^{-\sqrt{\kappa}}\right)\end{aligned}$$

to complete the proof. □

Roughly speaking, the above lemma states that if one wishes to have this probability be $2^{-\lambda}$, then the security parameter should be chosen as approximately $\kappa = O(\lambda^2)$. However, this guidance is worse than those in Figure 7 using Dickman's rho function directly.

H.2 Relationship between Restricted-Exponent and Algebraically-Restricted-Exponent Assumptions

We utilize the above result to formalize the relationship between the chosen-target, restricted-exponent assumption (Definition 4.3) and the chosen-target, algebraically-restricted-exponent assumption (Definition 4.4). Recall the only difference is that the exponent oracle performs additional checks in the latter game. We show that the two games are equivalent where the adversarial advantage difference is at most the probability upper bound that we proved in Theorem 7.1.

Theorem H.2. *If the $(\epsilon, t, \ell, \ell')$-chosen-target, restricted-exponent RSA inversion assumption (Def. 4.3) is true with $\mathcal{D}_N$ choosing uniformly random odd exponents from $[3, e_{\max}]$ where $e_{\max} \leq 2^{\kappa-2}$, then the $(\epsilon', t, \ell, \ell')$-chosen-target, algebraically-restricted-exponent RSA inversion assumption (Def. 4.4) is true with the same $\mathcal{D}_N$ where*

$$\epsilon' = \epsilon + \frac{\ell'^2}{x} + (2 \cdot \ell' \cdot \rho(\kappa - 2, \log x))$$

for some integer $x \geq 2$ and ρ is Dickman's function.

Proof. Towards a contradiction, suppose there exists an adversary $\mathcal{A}'$ running in time t' with ℓ and ℓ' queries to the decryption and exponent oracles respectively and advantage ϵ' for the chosen-target, algebraically-restricted-exponent game in Definition 4.4. We construct an adversary $\mathcal{A}$ for the chosen-target, restricted-exponent game in Definition 4.3. Essentially, $\mathcal{A}$ executes $\mathcal{A}'$ identically. We note that the advantage of $\mathcal{A}$ is identical to $\mathcal{A}'$ whenever the exponents output by the exponent oracle satisfy the checks. As the output of $\mathcal{D}_N$ is identical to H_{MD}, we can immediately apply Theorem 7.1 to obtain that the advantage of $\mathcal{A}$ satisfies

$$\epsilon \leq \epsilon' - \frac{\ell'^2}{x} - (2 \cdot \ell' \cdot \rho(\kappa - 2, \log x))$$

where we replace $|\mathbf{MD}|$ with ℓ' to limit the number of exponents exposed to the adversary. Note, we are free to pick any choice of integer $x \geq 2$ to minimize the right hand side of the equation. □

We can now use this relation to prove unforgeability of our blind RSA signatures with public metadata protocol directly from the chosen-target, restricted-exponent RSA inversion assumption.

Theorem H.3. *Suppose that $(H_{\mathcal{M}}, \mathsf{Verify}_{\mathcal{M}})$ correspond to full domain hash (FDH) message encoding. Assuming the $(\epsilon, t, \ell, |\mathbf{MD}|)$-chosen-target, restricted-exponent RSA inversion assumption (Definition 4.3) with exponents in $[3, e_{\max}]$ such that $e_{\max} = 2^{\kappa-2}$ and the random oracle model, then $\mathsf{BlindRSA}_{\mathsf{MD}}$ is $(\epsilon_F, t_F, \ell_F)$-strong one-more unforgeable (Definition 2.2) where*

- $\epsilon_F = \epsilon + 2^{-\kappa+2\log q_{\mathcal{M}}} + \frac{|\mathbf{MD}|^2}{x} + (2 \cdot |\mathbf{MD}| \cdot \rho(\kappa - 2, \log x))$ *for any integer $x \geq 2$*
- $t_F = t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$
- $\ell_F = \ell$

and the adversary $\mathcal{A}$ runs in time at most t_F and makes at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle and hash functions $H_{\mathcal{M}}$ and H_{MD} respectively.

Proof. First, we reduce the unforgeability game to the chosen-target, algebraically-restricted-exponent game in Theorem 6.1. Afterwards, we reduce to the final chosen-target, restricted-exponent game using Theorem H.3. □

We note that one could also plug Theorem H.1 into the above theorem to obtain the bound on adversarial advantage as

$$\epsilon_F = \epsilon + 2^{-\kappa+2\log q_{\mathcal{M}}} + O\left(2^{-\sqrt{\kappa}}\right).$$

I Survey of Cryptography Libraries for Production Usage

In this section, we survey the availability of various cryptographic operations across different libraries that are production ready (including those that are implemented to be specifically resistant to side-channel attacks). In particular, we will avoid any experimental and/or research libraries with explicit disclaimers that their implementations of cryptographic operations is not ready for production usage.

For our analysis, we will focus on the availability of the necessary cryptographic operations for our protocol (presented in this paper) as well as those based on pairings [56]. With respect to our protocol, we will focus on RSA operations with respect to large exponents. For pairings, we will simply analyze whether the cryptographic library exposes the necessary pairing-friendly curves for implementing [56]. As a caveat, we note our protocol also requires the additional cryptographic operations of generation safe primes during key generation that may not be exposed by some cryptographic libraries. We ignore safe prime generation because it is done exclusively by the signer in an offline manner (thus, the efficiency of safe prime generation is less important and the operation is less subject to side-channel attacks). Furthermore, it is not complex to implement safe prime generation using standard prime generation along with standard primality tests (both of which are available in every cryptographic library that we surveyed).

To choose production ready libraries, we consider several libraries that have active FIPS 140-2 certifications with current reviews for FIPS 140-3 validations (see [24] for comprehensive list). We caveat that not all cryptography libraries that are suitable for production usage have FIPS certifications. For example, we will include the ACML library [59] that supports pairings that was built for production usage without FIPS cerifications. Nevertheless, we believe this is a reasonable methodology for determining libraries whose cryptographic implementations are aiming to be usable in production systems. For our analysis, we will consider the ACML [59], BoringSSL [33], OpenSSL [45], wolfCrypt [62], BouncyCastle [58] and NSS [43] libraries. We present our findings in Figure 14 about the availability of RSA operations with large exponents and pairing-friendly curves.

RSA Operations with Large Exponents. We note that both BoringSSL and OpenSSL have the necessary public exposed functions to enable our protocol (our implementation [7] is built using BoringSSL). The required RSA operations are available for wolfCrypt

Library	RSA with Large Exponents	Pairing-Friendly Curves
BoringSSL [33]	✓	×
OpenSSL [45]	✓	×
wolfCrypt [62]	✓	×
BouncyCastle [58]	×	×
NSS [43]	✓	×
ACML [59]*	✓	✓

Figure 14: Comparison of the availability of RSA with large exponent operations and pairing-friendly curves in various cryptographic libraries suitable for production usage. Amongst this group, we note ACML is the only library without FIPS certifications.

that is also a derivative of OpenSSL (and we expect this to be true for most OpenSSL derivatives). Similarly, both the NSS and ACML libraries also offer the necessary RSA operations. We note that the necessary public functions are not available in BouncyCastle, but the required implementations exist (and would only need to be exposed to enable this implementation). This is not true for pairings where no implementation exists in BouncyCastle.

Pairing-Friendly Curves. In general, we can see that many of the cryptographic libraries ready for production usage do not support pairing-friendly curves (except ACML [59] that does not have FIPS certifications). This highlights the currently limited availability of pairing-based cryptography for production systems. We do caveat that pairings are available in many experimental and/or research-oriented library including Arkworks [9], CIRCL [21], NCC Group's implementation of BLS12-381 [47] and RELIC [51]. However, all of these have explicit disclaimers that the implementations should not be used in production. Additionally, we note there has been recent interest in pairing-based cryptography in IRTF [14] that may lead to wider availability of pairings in the future.