No evidence of communication: Off-The-Record Protocol version 4

Ola Bini
Centro de Autonomia Digital
ola@autonomia.digital

Sofía Celi
Centro de Autonomia Digital
sofia@autonomia.digital

1. INTRODUCTION

Cryptography is commonly used to secure private communications over the Internet. One way is to try to protect casual personal conversations, in a way that mimics the idea of a "casual" real-world conversation. However, in practice this does not work, since communications are only protected by long-lived encryption and signature keys, which are subject to compromise. In order to tackle this problem, the Off-The-Record protocol (OTR) was created [1], as a way to achieve two core properties: perfect forward secrecy and deniability (sometimes also referred to as repudiation). It is a protocol that uses encryption to hide the contents of a conversation and to protect against future compromises while also providing deniability, so that communications remain personal and do not provide proof of authorship.

OTRv4 [2] is the 4th version of this cryptographic protocol. It provides end-to-end encryption —information is sent over a network in such a way that only the recipient and sender can read it. It also provides offline deniability —transcripts provide no evidence even when long-term key material is compromised— and online deniability —no outsider can obtain evidence even when interactively colluding with an insider. Additionally, it provides participation deniability —transcripts provide no evidence that a specific party was part of the conversation. These properties come as a consequence of the Deniable Authenticated Key Exchanges (DAKEs) used in the protocol [3]. As the 4th version of OTR can handle both online and offline conversations, these latter properties depend on the type of conversation used. Online conversations provide both online and offline deniability for both participants in a conversation; while offline conversations provide offline deniability for both participants in a conversation, but only online deniability for the initiator of a conversation. Even with these limitations, the 4th version of the OTR protocol provides stronger deniability than all current secure messaging protocols in use, such as OTRv3 and Signal.

Since OTRv4 is designed to be used in the current messaging communication environment, it supports an out-of-order network model and provides different modes in which it can be implemented. The goal of the OTRv4 design is to create a version of the OTR protocol that does not only provide what OTR was created with; but to provide properties that apply to the current communication environment, to use up-to-date cryptographic primitives, and to define a protocol that can be used in the real world.

2. DESIGN OVERVIEW

One of the core properties of the current secure messaging protocols is deniability. This property can be defined as a way by which participants in a conversation can plausibility deny having participated in a conversation with each other, or the contents of it, as no cryptographic evidence of it can exist. However, the current secure messaging tools only allow for limited deniability, where the privacy and security of the participants engaging in a conversation can be compromised. With this in mind, we designed OTRv4 as an alternative to these messaging tools by providing strong deniability. The fundamental way by which we provide this is by using two DAKEs (one for online messaging and one for offline messaging): DAKEZ and XZDH, as defined by Unger et al [3].

While designing OTRv4, we also wanted to provide stronger confidentiality, which means that only the participants of a conversation are able to read the messages exchanged on it. Contrary to what other tools like Pretty Good Privacy (GPG) do, we use short-lived encryption/decryption keys, in the same way as the Double Ratchet Algorithm does [4]. This gives us backwards secrecy, forward secrecy and post-compromise security, as the compromise of both long-lived and short-lived key material does not compromise the confidentiality of neither past nor future messages. As each message will be encrypted and decrypted with one key only, message authentication and integrity verification can happen on a per message basis. This means that Message Authentication Codes (MAC) —a function computed over a message that takes a secret MAC key generated from the per-message encryption/decryption key— is only generated by the person with the ability to compute the correct keys. However, a MAC cannot provide non-repudation, as a participant is able to deny at a later time the authorship of the message signed —since both parties to the conversation have access to the MAC key. This means that either party could have generated this MAC, but the participant in a conversation is always assured that it came from the other party.

As we see, then, the purpose of OTRv4 is to provide an up-to-date protocol that provides no convincing cryptographic evidence that a conversation took place. Anyone could have modified or sent a message in this conversation. This idea also makes it possible for anyone watching a conversation that uses OTR to be able to forge it, or its contents. To achieve this property -known as forgeability- we use XSalsa20 [5]. Since this is a malleable encryption scheme, it makes it possible to alter messages.

3. LIMITATIONS

OTRv4 is a protocol that aims to be usable in the current secure messaging environment. To achieve this, it only uses available cryptographic primitives, which means that it uses primitives that have been implemented. As it aims to be an efficient and effective protocol, instead of using Diffie-Hellman key exchange, it uses

Elliptic Curve Diffie-Hellman key exchange, as the latter achieves the same security level as the first while using a shorter key length.

Additionally, OTRv4 does not take advantage of any quantum resistant algorithm, as their current implementations are not ready enough to be widely used. The design does include, however, what we call "weak quantum resistance": by periodically mixing in a 3072-bit Diffie-Hellman exchange in the double ratchet, we can guard for the possibility of quantum computers being able to break ECDH keys significantly faster than DH keys (because of the bit-size differences).

4. IMPLEMENTATION

Although there exist many cryptographic primitives, protocols and ideas, very few have been implemented. The idea of the OTR protocol and the new version of it is to give users a deniable and secure communication that can be used in the real world. The only way this can be proven is by showing how the protocol actually achieves what it claims. Implementing a proof-of-concept of it has several benefits; but it is not enough. The current secure messaging environment calls for proposals that can be used in real communication as soon as possible, as the privacy and security of users all over the world has already been compromised. For these reasons, we are developing a production ready implementation of OTRv4 in C [7], by using production-ready cryptographic primitives and libraries.

In our current work, we take into account some security measures. As we use elliptic curve cryptography, we check that the generated keys are not the identity-element, as this will make our implementation vulnerable to small sub-group [8] or invalid-curve attacks [9]. We use the Ed448-Goldilocks elliptic curve [11]. Furthermore, cryptographic operations use constant time operations [10], and we check for this by using the tool ctgrind [12]. We also check for initialized memory, memory leaks and double freeing with the programming tool valgrind [13]. We check race conditions in multi-threads with helgrind [14] and drd [15].

We are implementing the generic C library as a plugin for IM desktop clients, so that it can be used in real chat conversations.

5. CONCLUSION

As society becomes more and more dependent on electronic communication, protocols and tools that preserve privacy and provide security for its users are desperately needed. This new version of OTR, version 4, aims to provide those properties, as it achieves both confidentiality of the messages exchanged in a conversation, and deniability of both the transcripts and participation.

We are implementing the new version of the protocol as a library and plugin for IM desktop clients, as a production-ready tool and as a reference for implementations of the protocol in other programming languages.

6. REFERENCES

[1] Borisov, N., Goldberg, I., and Brewer, E. 2004. Off-the-Record Communication, or, Why Not to Use PGP. *Workshop on Privacy in the Electronic Society.*

[2] OTRv4 specification. https://github.com/otrv4/otrv4/blob/master/otrv4.md

[3] Goldberg, I. and Unger, N. 2016. Improved Strongly Deniable Authenticated Key Exchanges For Secure Messaging. Technical Report. University of Waterloo.

[4] Marlinspike, M. and Perrin, T. 2016. The Double Ratchet Algorithm. https://signal.org/docs/specifications/doubleratchet/

[5] Bernstein, D. 2008. Extending the Salsa20 nonce. *The University of Illinois at Chicago.*

[6] Diffie, W. and Hellman, M. 1976. New directions in cryptography. *Communications of the ACM.*

[7] Libotr-ng. https://github.com/otrv4/libotr-ng

[8] Antipa, A., Brown, D., Menezes, A., Struik, R., and Vanstone, S. Validation of Elliptic Curve Public Keys. *Public Key Crytography.*

[9] Hoon, C. and Joon, P. A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Sub-group. *Advances in Cryptology-CRYPTO '97.*

[10] Pornin, T. Why Constant-Time Crypto? https://www.bearssl.org/constanttime.html

[11] Hamburg, M. 2015. Ed448-Goldilocks, a new elliptic curve. *NIST ECC workshop.*

[12] Ctgrind. *https://github.com/agl/ctgrind*

[13] Valgrind. http://valgrind.org/

[14] Helgrind. http://valgrind.org/docs/manual/hg-manual.html

[15] DRD. http://valgrind.org/docs/manual/drd-manual.html