Secure Messaging, Riposte, Vuvuzela

Scribe: Arjun Baokar

October 27, 2015

1 Secure Messaging

There are three major parts to creating a secure messaging system:

1. **Trust establishment:** ensure that you know who you're talking to
2. **Conversation security:** ensure that other people can't read your conversation
3. **Transport security:** ensure that metadata is hidden

1.1 Threat Model

The threat model assumes that an attack controls all, or at least part of the network over which the communication is happening. Furthermore, the attacker can be passive or active.

1.2 Security Goals

- **Confidentiality:** An attacker should not be able to see (decrypt) the conversation without owning the private keys.
- **Non-repudiation/repudiation:** A participant in the conversation should be able to prove that they did or didn't send a message that someone else claims they sent. Depending on what the messaging system is trying to ensure, it may value repudiation over non-repudiation, or vice versa.
- **Integrity:** An attacker cannot change or alter the message without being at least detected.
- **Forward Secrecy:** If a long-term key is compromised, the attacker cannot decrypt conversations before the key compromise.
- **Backward Secrecy:** If a long-term key is compromised, the attacker cannot decrypt conversations after the key compromise.
- **Authentication:** A participant knows who they are communicating with.
- **Anonymity:** People not participating in the conversation cannot determine who is participating in the conversation.

Secure messaging schemes are judged based on the *security*, *usability*, and *ease of adoption* of the scheme. Additionally, good performance and availability is also desired.

1.3 Trust Establishment Schemes

Trust establishment involves the exchange and setup of long-term keys. Below are some schemes that attempt to set this up, along with a discussion of their pros and cons.

1.3.1 Opportunistic Encryption

Alice and Bob do a Diffie-Hellman key exchange. Alice and Bob both thus end up with a g^{ab} secret key.

- **Pros:** great usability, easy to adopt, secure if the setup stage is not compromised
- **Cons:** no authentication since anyone can pretend to be a participant during setup, weak against MITM attacks for the same reason

1.3.2 Trust on First Use (TOFU)

This is a slightly modified version of opportunistic encryption. Alice and Bob once again do a Diffie-Hellman key exchange on their first interaction, but they maintain a database of user: public_key pairs. On subsequent key exchanges, they check their database to make sure that their partners have the same keys as before, avoiding subversion of authentication on subsequent messages. Thus, you only need to trust the first exchange.

- **Pros:** great usability, easy to adopt, narrows the window of an attack (the attacker must trick them on the first key exchange)
- **Cons:** authentication issues on first key exchange, makes key revocation very difficult

1.3.3 Key Fingerprinting

Alice and Bob talk on a separate trusted channel, such that they have both shared some secret that they know (g^{ab} in this case). They then send each other the hash of the secret, $H(g^{ab})$, and confirm that they both have the same secret. This protects against MITM attacks, since the MITM would not send them both the same secret.

To simplify the checking step, the hash is often linked to a physical word or phrase, so it is human readable. The participant can even enter the message he gets from Alice into his client, which will check it for him.

- **Pros:** detects MITM attacks unlike the previous schemes
- **Cons:** requires another trusted channel, problem for usability

1.3.4 Authority-Based Trust

The public keys are signed by some trusted authority (like Certificate Authorities). Alice and Bob then send the authority-signed public keys to each other. This is like the current TLS model.

- **Pros:** great usability
- **Cons:** hard to deploy, need to trust CA (which may not always be trustworthy)

1.3.5 Keybase

A user can register with Keybase by providing their public key and some proof of ownership on the key. They can then associate social networks (Twitter, Facebook, etc) with their public key. People looking for their public key can look them up by social media username.

- **Pros:** great usability, easy to deploy
- **Cons:** cryptography in browser

1.4 Conversation Security

Conversation security's goal is to hide the content of conversations. Once a shared symmetric key has been established, clients will encrypt and provide a MAC (Message Authentication Code) for their messages.

1.5 Transport Security

The goal of transport security is to provide anonymity by hiding the metadata of the conversation (eg. who and when users talk to). There should be both anonymity and unlinkability (2 messages should not be able to be linked to the same conversation).

1.5.1 Strawman: Trusted Party

Everyone sends messages to a trusted third part, which forwards their respective messages to all of the users using the service.

- **Pros:** fast (low latency), easy to use, easy to adopt
- **Cons:** central point of attack, must trust third party to not be malicious

1.5.2 Peer to Peer Network

Each user has a distributed hash table of peers, which maps the peer's IP to the set of usernames it serves. If Alice wants to message Bob, she will look up the peer that is responsible for serving Bob, and send that peer the message for Bob. The peer will then forward the message to Bob.

- **Pros:** great usability, no central point of attack
- **Cons:** Sybil attacks, availability can be an issue with peers, peers can be compromised, monitoring network links can affect anonymity

1.5.3 Onion Routing (Tor)

Send messages over Tor. This provides the same security guarantees as onion routing. See Tor notes for more details.

1.5.4 Dining Cryptographer Network (DC Net)

Create a cycle of clients that all send messages directly to the next person in the cycle. The sending is divided into "rounds" or "epochs", which is essentially a set interval where everyone must pass a message onto the next peer.

If Alice wants to send a message to Bob, she encrypts a message with Bob's public key and sends it to the next person in the cycle.

- **Pros:** strongest security guarantee of aforementioned systems (even an attacker with a global view cannot deanonymize)
- **Cons:** high bandwidth, fixed latency, no real-time, vulnerable to disruptors (clients that don't participate properly)

2 Riposte

Riposte is a system based on the concept of DC Nets that works as an anonymous messaging board. It does not provide real-time communication, so messages must be latency tolerant. It provides protection from traffic analysis atacks and strong disruption resistance.

2.1 Threat Model

It assumes that the clients are untrusted and can perform active attacks. It assumes that at least one server involved in the system is not malicious, and servers do not collude.

2.2 Basic Strawman Example

Each server views the database as a L-bit long bitstring, and clients can issue write requests to the servers. The server collects all client requests over a time epoch, and publishes all requests at the end of the epoch. All client write requests within a time epoch constitute the anonymity set.

Each server client submits a different value bitstring to each server. For example, in the basic strawman example of two servers A and B:

- Client $->$ A: L-length bitstring r
- Client $->$ B: $r \oplus e_i$ where i is the position in the database it wants to write to

2.3 Contributions

The actual Riposte model fixes many shortcomings in the strawman model.

- To avoid collisions, Riposte servers store $(m_a, m_a^2, ..., m_a^k)$ instead of just the message m_a from each client to allow for the reconstruction of up to k collisions.
- Riposte uses PIR (private information retrieval) techniques, such as DPF (distributed point function).
- Riposte uses secure multiparty computation to stop disruption attacks.

2.4 More Details

DPF is a piecewise function which essentially creates a non-zero value for a given value p, while creating a zero value for all other values. Stated otherwise: $DPF(x)$:={nonzero if $x = p$, 0 if $x \neq p$}.

Riposte also creates an Eval method for each server. The Eval method for an individual server does not reveal the messages that a users submitted during that time epoch, while the sum of the Eval methods for each server do. Thus, to release the messages at the end of the epoch, servers sum over what they know. Stated otherwise: $\sum Eval(k, l') = \sum m'$.

Another insight that Riposte provides is that using a random generator G in its Eval functions stretches the randomness within its anonymity set.

3 Vuvuzela

Vuvuzela's goal is to protect metadata and ensure anonymity. It involves the use of a trusted server in a communication network. Furthermore, it also provides a much higher bandwidth than previous solutions. Vuvuzela can scale to over 1 million users sending a total of around 68,000 messages per second.

3.1 Threat Model

The threat model assumes that clients cannot be trusted, the attacker can observe all network traffic, and that there is at least one honest Vuvuzela server.

3.2 Basic Implementation

Vuvuzela creates a scheme involving "dead drops" on their servers. Each participant in a conversation is assigned the same dead drop (chosen randomly) . To send data, the client leaves encrypted messages in that dead drop, which is housed on the Vuvuzela servers. When receiving messages, the other client picks up the messages from the dead drop. Each round, the protocol chooses a new dead drop pseudo-randomly. All accesses to these dead drops are hidden.

It also implements a dialing protocol, which allows people to set up a communication with one another, identifying each other via long-term public keys.

Every client must partake in the message exchange. At the end of each round, all of the dead drop messages are passed to each server, and each server mixes (permutes) the messages it has to make it harder to figure out what the messages were. If at least one server is honest and permutes the data as it should, then other dishonest servers cannot figure out the message or deanonymize the users in conversations.

A dishonest server can successfully choose to not forward messages (a Denial of Service attack), but cannot gain any information to compromise anonymity of a communication.

3.3 Contributions

Vuvuzela's protocols reveal only specific information to an attacker, such as the total number of participants in a conversation (and conversely, the total number not involved in one). Each

protocol only has a very controlled amount of information leakage, which is further obscured using differential privacy techniques. This provably hides which participants are communicating.

Vuvuzela also provides a lot more scalability than other secure messaging systems established before it, scaling to 1 million users and 68,000 messages per second with 37 second end-to-end latency.