A Time-Lapse Cryptography Service

Christopher Thorpe
and Michael Barrientos
General Cryptography
{cat,michael}@general-cryptography.com

Prof. Michael O. Rabin
Harvard University
Cambridge MA 02138
rabin@seas.harvard.edu

Server-Side Key Construction and Distribution

A consort of servers use *distributed key generation* to construct components of an ElGamal **public** and **secret** key pair associated with a *fixed decryption time*.

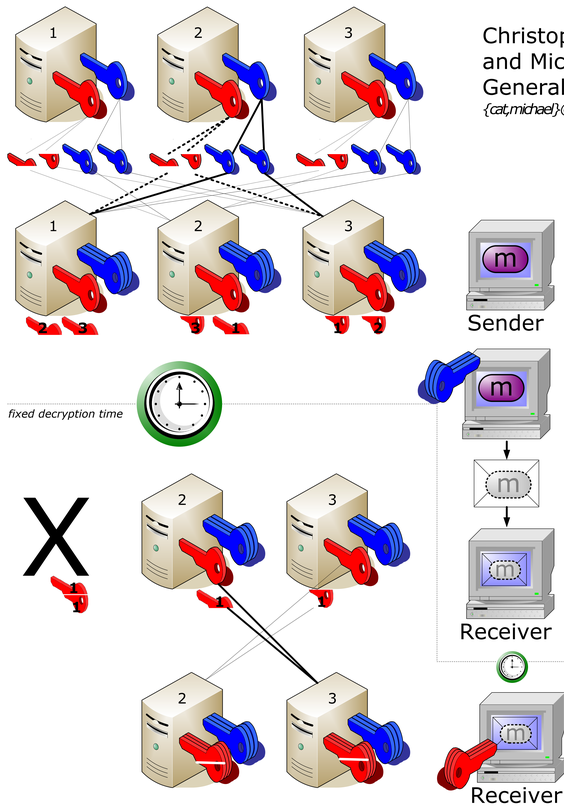
The servers broadcast the **public** key components, and use *verifiable secret sharing* to share their **secret** key components with other servers.

The servers construct the **public** key from the broadcast components and make it available for download.

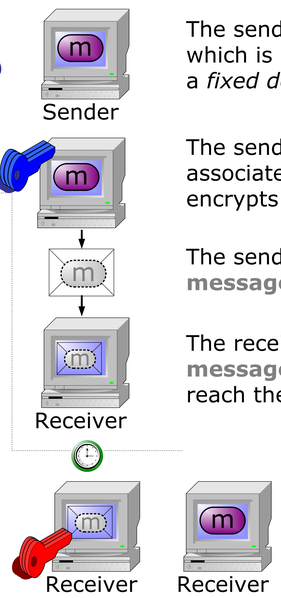
The servers wait for the *fixed decryption time,* as reported by a trustworthy **clock**.

At the fixed decryption time, the servers broadcast their **secret** key components and shares of other servers' **secret** key components. They also reconstruct the components of any missing servers' keys.

The servers assemble the components, reconstructing the **secret** key. They publish the **secret** key by making it available for download.



Client-Side Secure Communication



The sender creates a **message** m which is intended to be published at a *fixed decryption time.*

The sender downloads the **public** key associated with the desired *time*, and encrypts the **message**.

The sender sends the **encrypted message** to the receiver.

The receiver accepts the **encrypted message** and waits for the **clock** to reach the *fixed decryption time*.

The receiver downloads the **secret** key associated with the desired *time*, and decrypts the **message**.

Introduction and Applications

Time-Lapse Cryptography (TLC) is a formalized method of securely sending a message into the future: messages are encrypted with public keys whose corresponding secret keys are unknown to any party until they are reconstructed at a specified future time. A group of servers jointly create public keys, only a majority of which is required to reconstruct the secret keys later.

The TLC Service is useful in any application where data are to be committed to for guaranteed future disclosure. The service provides an enhanced cryptographic commitment, with the following properties:

- **binding**: a sender cannot change a value committed to
- **hiding**: a recipient cannot learn anything from a commitment
- **non-repudiation**: a sender cannot repudiate a commitment

One example of such an application is a cryptographic auction, where bid information is protected from auctioneers until bidding is closed but accessible to them afterward.

Assumptions

Decisional Diffie-Hellman
We employ ElGamal encryption and digital signatures and a variant of Feldman's verifiable secret sharing, all of which are secure under the Decisional Diffie-Hellman (DDH) assumption.

Clocks
We assume that the TLC servers have accurate internal clocks and access to multiple external, trustworthy clocks for coordinating reconstruction of secret keys.

Server security
We assume that the TLC servers that hold private data can be made secure from network and physical attacks. We assume that that the TLC servers do not collude and that it is impractical to compromise a sufficient number of them to gain data about the secret key information prematurely.

Technologies We Employ

- **Secret Sharing:** We implemented Shamir's information-theoretically secure polynomial threshold secret sharing for sharing secret key components. We also employ a secure variant of Feldman's verifiable secret sharing for greater security.
- **Cryptography:** We use ElGamal cryptography for TLC keys and digital signatures. We also implemented a wrapper to encrypt arbitrary data streams using AES (Rijndael) with random symmetric private keys secured with TLC.
- **Communications:** We use secure communication channels over SSH and PKI with cross-signed ElGamal digital signature keys for all TLC servers.
- **Distributed Architecture:** We intend new TLC servers to be distributed so that no corporation or government can force early reconstruction of any key.
- **Systems:** We implemented TLC in Erlang 5.6.5 on servers running Debian 4.0 Linux on Intel 2.0GHz quad-core Xeon processors.
- **Clocks:** To determine reconstruction times for secret keys, we compare servers' internal clocks with trustworthy Network Time Protocol (NTP) servers around the world. Discrepancies that are too large trigger an alert.