Off-the-Record Messaging: Useful Security and Privacy for IM

Digital Security Seminar
Carleton University

Ian Goldberg
Cryptography, Security, and Privacy (CrySP) Research Group
University of Waterloo
<i iang@cs.uwaterloo.ca>

24 April, 2007
The challenge

Researchers have had a hard time getting their work in security and privacy technologies to benefit real people.

- It's hard to use!
- It's hard to get!
- It doesn't work!
The goal

- At the end of the day, what matters is that the technologies we produce actually improve people's lives in some way!

- Our goal is to create what we call **Useful Security and Privacy Technologies**.
Useful Security and Privacy

• There are four major aspects to such technologies:
  – Usability
  – Deployability
  – Effectiveness
  – Robustness

• We'll quickly look at what these all mean.
Usability

• Usability is the best known of these properties.

• We not only mean it in the sense of user interfaces, and “usable security”, however.

• For example, if a privacy technology causes your web browsing to slow to an unacceptable crawl, that's an unusable technology.
Deployability

• But making a technology easy to use isn't enough!

• It also has to be *reasonable* to use.
  – If users have to change their:
    • operating systems
    • web browsers
    • instant messaging clients
  – then they won't want to use your technology.
Effectiveness

• Of course, even assuming the users *have* the technology, it needs to do them some good.

• All too often, we see that many proposed, and even widely deployed, security systems have major flaws.
  – Peer review, analysis
  – Not only of the design, but also the implementation
Robustness

• Many times, security technologies work only so long as everything goes “according to plan”.
  – Small deviations from the assumptions made by designers can cause the systems to fail catastrophically!

• But:
  – Users forget passwords
  – Their computers are compromised by malware
  – They misunderstand security-relevant messages
  – They fall victim to phishing attacks
  – etc.
An example

- Alice and Bob want to communicate privately over the Internet.

- Generous assumptions:
  - They both know how to use PGP
  - They both know each other's public keys
  - They don't want to hide the fact that they talked, just what they talked about
Threat model

Alice

The Internet

Bad Guys

Bob
Solved problem

• Alice uses her private signature key to sign a message
  – Bob needs to know who he's talking to
• She then uses Bob's public key to encrypt it
  – No one other than Bob can read the message
• Bob decrypts it and verifies the signature

• Pretty Good, no?
Threat model

Alice

Bad Guys

The Internet

Bob
Plot twist

● Bob's computer is stolen by “bad guys”
  – Criminals
  – Competitors
  – Subpoenaed by the RCMP

● Or just broken into
  – Virus, trojan, spyware, etc.

● All of Bob's key material is discovered
  – Oh, no!
The Bad Guys can...

- Decrypt past messages
- Learn their content
- Learn that Alice sent them
- And have a mathematical proof they can show to anyone else!

- How private is that?
What went wrong?

• Bob's computer got stolen?

• How many of you have never...
  – Left your laptop unattended?
  – Not installed the latest patches?
  – Run software with a remotely exploitable bug?

• What about your friends?
What really went wrong

• PGP creates lots of incriminating records:
  – Key material that decrypts data sent over the public Internet
  – Signatures with proofs of who said what

• Alice had better watch what she says!
  – Her privacy depends on Bob's actions
Casual conversations

• Alice and Bob talk in a room

• No one else can hear
  – Unless being recorded

• No one else knows what they say
  – Unless Alice or Bob tells them

• No one can prove what was said
  – Not even Alice or Bob

• These conversations are “off-the-record”
We like off-the-record conversations

• Legal support for having them
  – Illegal to record conversations without notification

• We can have them over the phone
  – Illegal to tap phone lines

• But what about over the Internet?
  – Instant Messaging (IM) “feels” like a personal conversation
  – If only it were so!
Crypto tools

• We have the tools to do this
  – We've just been using the wrong ones
  – (when we've been using crypto at all)

• We want perfect forward secrecy

• We want deniable authentication
Perfect forward secrecy

• Future key compromises should not reveal past communication
• Use a short-lived encryption key
• Discard it after use
  – Securely erase it from memory
• Use long-term keys to help distribute and authenticate the short-lived key
Deniable authentication

• Do not want digital signatures
  – Non-repudiation is great for signing contracts, but undesirable for private conversations

• But we do want authentication
  – We can't maintain privacy if attackers can impersonate our friends

• Use Message Authentication Codes (MACs)
MAC operation

Alice

Bob

Data

MAC

MK

MK

Data

MAC

=?
No third-party proofs

• Shared-key authentication
  – Alice and Bob have the same MK
  – MK is required to compute the MAC

• Bob cannot prove that Alice generated the MAC
  – He could have done it, too
  – Anyone who can verify can also forge

• This gives Alice a measure of deniability
Off-the-Record Messaging protocol

• Rough sketch of protocol
  – Details on our web page
• Assume Alice and Bob know each other's public keys
  – These keys are long-lived, but we will only use them as a building block
  – Use “ssh-style” approach: users are warned the first time they see a particular key; keys are verified thereafter
Step 1: Authenticated Key Exchange

- We use a variant of the SIGMA protocol as the AKE.
  - The same protocol that's used in IKE (part of IPsec)
- Use plain (unauthenticated) Diffie-Hellman to set up a channel
- Sign a MAC on fresh data to prove your identity and that you know the shared secret
Details of SIGMA

\[ A : \quad x \in_R 2^{320}, r \in_R 2^{128} \]

\[ A \rightarrow B : \quad E_r (g^x), \text{Hash}(g^x) \]

\[ B : \quad y \in_R 2^{320} \]

\[ B \rightarrow A : \quad g^y \]

\[ A : \quad s = (g^y)^x \]

\[ A \rightarrow B : \quad r, E_{h_1(s)} (A, \text{sig}_A (\text{MAC}_{h_2(s)} (g^x, g^y, A))) \]

\[ B : \quad s = (g^x)^y \]

\[ B \rightarrow A : \quad E_{h_3(s)} (B, \text{sig}_B (\text{MAC}_{h_4(s)} (g^y, g^x, B))) \]
Step 2: Message transmission

- Compute \( E_K = \text{Hash}(s) \), \( M_K = \text{Hash}(E_K) \)

\[
A \rightarrow B : \quad E_{E_K}(M), \ MAC_{M_K}(E_{E_K}(M))
\]

- The encryption is AES in Counter mode
- Bob verifies MAC using \( M_K \), decrypts \( M \) using \( E_K \)
- Confidentiality and authenticity are assured
Step 3: Re-key

- As often as possible, Alice and Bob pick new x', y'

  \[ A \rightarrow B : \ g^{x'}, \text{MAC}_{MK} \left( g^{x'} \right) \]
  \[ B \rightarrow A : \ g^{y'}, \text{MAC}_{MK} \left( g^{y'} \right) \]
  \[ A, B : \ s' = g^{x'y'} \]

- Compute EK' = Hash(s'), MK' = Hash(EK')

- Alice and Bob securely erase s, x, y, and EK
  - Perfect forward secrecy
Step 4: Publish MK

- Alice and Bob do not need to forget MK

- They no longer use it for authentication

- In fact, they publish the old MK along with the next message
  - This lets anyone forge messages, but only past ones
  - Provides extra deniability
Deniability

• OTR offers many layers of deniability:
  – During the AKE:
    • The signature only proves that Alice has used OTR at some point in the past, not even with any particular person
  – During message transmission:
    • Bob can't prove to Charlie that the messages he gets are coming from Alice, even though he himself is assured of this
  – After the fact:
    • AES Counter mode allows messages to be altered once the MAC key is published
    • Entire new messages, or whole transcripts, can in fact be forged
• It's not more deniable than plaintext, of course!
Using these techniques

- Using these techniques, we can make our online conversations more like face-to-face “off-the-record” conversations.

- But there's a wrinkle:
  - These techniques require the parties to communicate *interactively*
  - This makes them unsuitable for email
  - But they're still great for instant messaging!
Off-the-Record Messaging

- Off-the-Record Messaging (OTR) is software that allows you to have private conversations over instant messaging, providing:
  - Encryption
    - Only Bob can read the messages Alice sends him
  - Authentication
    - Bob is assured the messages came from Alice
Off-the-Record Messaging

• Perfect Forward Secrecy
  – Shortly after Bob receives the message, it becomes unreadable to anyone, anywhere

• Deniability
  – Although Bob is assured that the message came from Alice, he can't convince Charlie of that fact
  – Also, Charlie can create forged transcripts of conversations that are every bit as accurate as the real thing
Off-the-Record Messaging

• Availability of OTR:
  – It's built in to Adium X (a popular IM client for OSX)
  – It's a plugin for gaim (a popular IM client for Windows, Linux, and others)
    • With these two methods, OTR works over almost any IM network (AIM, ICQ, Yahoo, MSN, etc.)
  – It's a proxy for other Windows or OSX AIM clients
    • Trillian, iChat, etc.
  – Third parties have written plugins for other IM clients
    • Miranda, Trillian, Kopete
Is OTR Useful?

• OTR is easy to use
  
  – The software automatically notices when Alice and Bob both support OTR, and automatically protects their conversations.
  
  – The IM servers just pass encrypted messages back and forth between Alice and Bob, unaware that anything unusual is going on.
Is OTR Useful?

• OTR is easy to deploy
  – You probably don't have to change your IM client to use OTR.
  – In fact, your IM client might support OTR already!
  – It's also part of many standard OS distributions.
Is OTR Useful?

• It works
  – Peer-reviewed design
  – Open-source implementation

• Robust against failures
  – Preserves security in the face of simple failures
  – Preserves deniability in the face of major failures
Is OTR Useful?

• OTR is a good example of a Useful Security and Privacy Technology.

• Tens of thousands of people are using OTR to protect their IM conversations.
Future directions

• More flexible key verification
  – When Alice first talks to Bob, she will be presented with a fingerprint
    B0A09015 B20564FB 71E1AFEE 8FC1A8F1 EEAA6379
  – How does she know if this is the correct one, or if there's a MITM?
  – She needs to determine this out-of-band, through some other authenticated channel
    • She can phone Bob, and have him read it to her (and recognize his voice)
    • Bob can put a PGP-signed message containing his OTR fingerprint on his web page
More flexible key verification

- Alice and Bob can arrange a shared secret, and compare them without revealing any information except whether the secrets match.
  - This way, they can meet before they've both installed OTR, and still be able to maintain security and privacy.
  - Alice and Bob each end up computing
    \[ w^{(s_A - s_B)} \]
    for a random value \( w \).
Private chats for groups

- What if more than two people want to have a private chat room?
  - What does "deniability" mean in this situation?
    - That you can claim you weren't part of the chat at all?
    - That you can claim someone else in the chat wrote a particular message?
  - Are people in the chat assured of the actual author of a message?
    - Or just that someone in the room said it?
  - Can you do it without effectively setting up person-to-person private conversations for each pair of people in the room?
Comparison to other systems

- gaim-encryption
  - Encryption and authentication
  - No deniability or perfect forward secrecy
  - Like PGP with signatures

- Trillian SecureIM
  - Encryption with perfect forward secrecy
  - No authentication at all

- SILC
  - Completely separate network
  - Share messages (securely) with SILC server, or
  - Pre-shared long-term secret, or
  - Peer-to-peer communication (hard with NATs)
For more information

• For more information about OTR, see our web page:

  http://otr.cypherpunks.ca/