Security Protocols

Network Security
mod-net-sec
University of Birmingham
Core ideas

• Do not re-invent the following:
  – Security protocols
  – Hash functions
  – Block ciphers
  – Modes of operation

• The community is littered with the bones of flawed proposals of all these

• It doesn't matter, even if you are a capable and experienced cryptographer

• Use peer-reviewed building blocks where possible
Two case studies

• Scatter chat
• WEP – 802.11b standard
ScatterChat is a secure instant messaging client designed for non-technical users who require secure and anonymous communications. Our typical end-users include human rights and democracy advocates operating in hostile territory. ScatterChat is also a valuable tool for anyone requiring secure communications.

It is based upon the Gain software, provides end-to-end encryption, integrated onion-routing with Tor, secure file transfers, and easy-to-read documentation.

ScatterChat's security features include resiliency against partial compromise, immunity from replay attacks, and limited resistance to traffic analysis... all reinforced through a pro-actively secure design.

Update {08/05/2006}: A note regarding the perfect forward secrecy terminology has been posted, along with an explanation to a common concern regarding the use of ECB mode.

Update {08/04/2006}: The Scatterchat-announce mailing list is now available for those wanting notifications of new versions.

Update {07/30/2006}: A frequently asked questions section is now up!

Update {07/29/2006}: A wiki for the User's Guide is now up for proofreaders to help out! Mailing lists are now available for volunteers as well.
ScatterChat's goals

• ScatterChat is similar to OTR with one exception: OTR automatically responds to a key-exchange request confirming the use of OTR

• This is a security problem when the use of encryption itself is considered suspicious

• ScatterChat was designed not to respond to requests from unknown users.

• What if ScatterChat were susceptible to a person-in-the-middle attack?

• Another goal of ScatterChat is **Perfect Forward Secrecy**
Perfect forward secrecy

• A standard goal of many communication protocols
• PFS means that past data is not compromised upon the loss of long-term secrets
• Standard way to achieve this:
  – Participants use Diffie-Hellman key-exchange to obtain a shared session key
DH key-exchange

Alice
- Common paint
- Secret colours
- Public transport
- Secret colours
- Common secret

Bob
- Common paint
- Secret colours

DH Key-exchange:

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secret</td>
<td>Public</td>
</tr>
<tr>
<td>a</td>
<td>p, g</td>
</tr>
<tr>
<td>a</td>
<td>p, g, A</td>
</tr>
<tr>
<td>a, s</td>
<td>p, g, A, B</td>
</tr>
</tbody>
</table>
Perfect forward secrecy

- A standard goal of many communication protocols
- PFS means that past data is not compromised upon the loss of long-term secrets
- Standard way to achieve this:
  - Participants use Diffie-Hellman key-exchange (with digital signature) to obtain a shared session key
  - Subsequent data is encrypted under the session key which is discarded at the end of the session
Class activity

• Analyse the ScatterChat design document and comment on how well it achieves its goals:
  – Resistance against the detection (of encryption)
  – Perfect Forward Secrecy
  – and of course, Traffic analysis resistance
Analysis of resistance to detection of encryption

• Simplified version of the protocol

Alice initiates a ScatterChat session:
\[ A \rightarrow B: S_{a_{pub}} \mathbin{||} E_{a_{pub}} \mathbin{||} \text{Sig}(S_{a_{priv}}, E_{a_{pub}}) \]

Bob verifies Alices' signature and confirms that he is willing to let Alice know that he supports ScatterChat:
\[ B \rightarrow A: S_{b_{pub}} \mathbin{||} E_{b_{pub}} \mathbin{||} \text{Sig}(S_{b_{priv}}, E_{b_{pub}}) \]

where:
- \( S_{x_{pub}} \): X's public and private signing keys respectively
- \( E_{x_{pub}} \): X's public encryption key
- \( \text{Sig}(k, d) \): Signature of data \( d \) under key \( k \)
Analysis of resistance to detection of encryption

• Note: No guarantee of freshness
• This allows Eve to carry out a person-in-the-middle attack
• Partial spoofing attack: Eve replays $\text{Sig}(Sa\_priv, Ea\_pub)$ to Bob who responds assuming it is Alice, hence revealing his use of encryption.
PFS analysis

• DH key exchange
• Each side generates a random session key component and encrypts it to the other party’s public key.
• These components are xor-ed to derive the session key.
• If both the private encryption keys are revealed, then the transfer of the session key components can be decrypted, hence revealing all previous conversations.
Traffic-analysis resistance

- ECB consistently encrypts the same plaintext block to the same ciphertext block so reveals patterns in the plaintext.

Electronic Codebook (ECB) mode encryption
Traffic-analysis
Block cipher modes

• CBC or CTR modes do not reveal patterns if IV is properly set.

Cipher Block Chaining (CBC) mode encryption
ECB consistently encrypts the same plaintext block to the same ciphertext block so reveals patterns in the plaintext.

Traffic analysis and Block cipher modes

Original image

Encrypted using ECB mode

Modes other than ECB result in pseudo-randomness
TA resistance in ScatterChat

- ScatterChat takes a different approach of inserting one random byte for every three bytes of plaintext.
- This introduces a 33% overhead and puts strain on the operating system entropy pool, but it was hoped to resist the weaknesses of ECB.
- Unfortunately, for significant message lengths, it fails to provide adequate protection. Why?
TA resistance in ScatterChat

- AES Block size is 128 bits
- For every block there is 32 bits of random data.
- the Birthday theorem
  [http://en.wikipedia.org/wiki/Birthday_paradox](http://en.wikipedia.org/wiki/Birthday_paradox) says that upon sampling with replacement from a pool of n samples, the chance of collision is 50% after roughly sqrt(n) samples have been drawn.
• So, after roughly $\sqrt{2^{32}} = 2^{16}$ or $\sim 64k$ blocks, the ciphertexts will repeat with high probability whenever the plaintexts are the same.
• This allows the attacker to find patterns in the messages
• This corresponds to 904kB of plaintext
• For the probability to be 1% it is only 109kB and reaches 99% after 2.3MB.
A cautionary tale

• Instead of inventing ScatterChat, the engineers could have achieved a similar result with GAIM, and official OTR plugin then configuring them to use Tor, albeit without secure file transfer.

• Further reading: “Off the record communication” http://www.cypherpunks.ca/otr/otr-wpes.pdf
Core ideas

- Do not re-invent the following:
  - Security protocols
  - Hash functions
  - Block ciphers
  - Modes of operation
- The community is littered with the bones of flawed proposals of all these
- It doesn't matter, even if you are a capable and experienced cryptographer
- Use **peer-reviewed building blocks** where possible
Next, WEP

• Before we go there, remember: use open source protocols!