Secure Channels

Christian Grothoff

Berner Fachhochschule

27.4.2018

Learning Objectives

- What are cryptographic protocols?
- Protocols for key exchange without public key cryptography
- Protocols for key exchange with public key cryptography
- What are secure channels?
- Terminology: Forward secrecy, future secrecy, asynchrony, repudiation
- Contemporary protocols for secure channels
- Attacks
- Modern secure channels

Protocols

- "A **protocol** is a series of steps, involving two or more parties, designed to accomplish a task."
- Everyone involved must know the steps in advance and agree to follow it.
- The protocol must be complete and unambiguous.
- For cryptographic protocols, it should not be possible to do more or learn more than *what is specified in the protocol*.

Dramatis Personae

- Alice, Bob, Carol and Dave
- Eve – Eavesdropper
- Mallory – Malicious active attacker
- Trent – Trusted arbitrator
- Walter – Warden
- Peggy – Prover
- Victor – Verifier

Attack Personae

- Eavesdroppers
- Passive cheaters
- Active cheaters
- Real-world adversaries – Mallory

Efficiency

- Number of steps in protocol
- Size of messages
- Conflict resolution cost:
  1. Involvement of trusted party (arbitrated protocols)
  2. Resolution by trusted party on dispute (adjudicated protocols)
  3. Self-enforcing protocols

Example: Symmetric Cryptography

1. Alice and Bob agree on a cryptosystem
2. Alice and Bob agree on a key
3. Alice encrypts plaintext with key
4. Alice sends ciphertext to Bob
5. Bob decrypts ciphertext and reads it

Problem

Alice has an item x, and Bob has a set of five distinct items $y_1, \ldots, y_5$. Design a protocol through which Alice (but not Bob) finds out whether her x equals any of Bob's five items; Alice should not find out anything other than the answer ("Yes" or "No") to the above question, and Bob should not know that answer. Your solution must always be correct, not just with high probability.

Key Establishment Security goals

The basic security goals of key establishment are:

- *Key secrecy:* Session keys must not be known by anyone else than Alice, Bob (and maybe some trusted third party). Mallory must not learn anything about session keys.
- *Authenticity:* One party can be assured about the identity of the other party it shares the session key with. That is, Alice knows that she has session key with Bob.
- *Freshness of keys:* Mallory must not be able to replay old session keys.

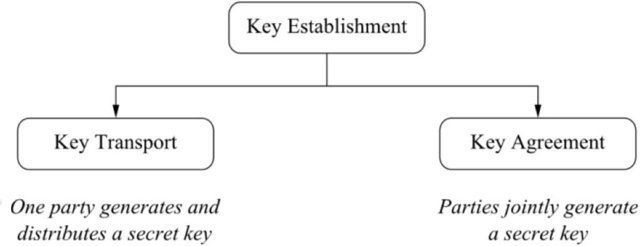
Protocols

- *Key establishment* is realized by using *protocols* whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use.
- Until now, we have been discussing non-interactive crypto primitives, in the following we look at crypto protocols.
- It is *even harder* to design secure protocols, than designing non-interactive primitives. In fact, there is a long list of protocols designed by famous (and not so famous) cryptographers that were found to be flawed.

Session keys

- Key establishment protocols result *in shared secrets* which are typically called (or used to derive) session keys.
- Ideally, a session key is an ephemeral secret, i.e., one whose use is *restricted to a short time period such as a single telecommunications connection (or session)*, after which all trace of it is eliminated.
- Motivation for ephemeral keys includes the following:
  1. To limit available ciphertext (under a fixed key) for cryptanalytic attack;
  2. To limit exposure, with respect to both time period and quantity of data, in the event of (session) key compromise;
  3. To avoid long-term storage of a large number of distinct secret keys by creating keys only when actually required;
  4. To create independence across communications sessions or applications.

Classification of key establishment methods



Chapter 13 of Understanding Cryptography by Christof Paar and Jan Pelzl

Private channels

- Let us informally refer to a *private channel* as an authentic and confidential channel.
  - Exchange of secret keys on a USB stick
  - Pre-installation of keys on a company laptop
- Symmetric key distribution is impossible without private channels.
- Private channels are, loosely speaking, “complicated”, “inefficient”, “expensive”.
- The goal in the following is to:
  - *Reduce the number* of private channels required to exchange keys.
  - Use an *initial private channel* today to exchange a secret key that they may use *tomorrow for establishing a secure channel over an insecure link*.

Storytime

Once upon a time...

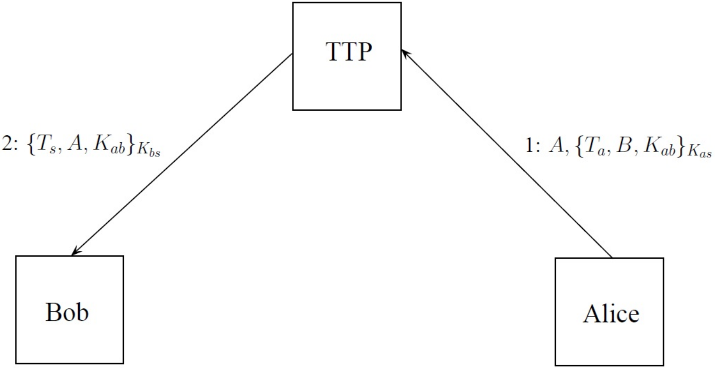
Neumann-Stubblebine

1. Alice sends A, R_A to Bob.
2. Bob sends $B, R_B, E_B(A, R_A, T_B)$ to Trent, where T_B is a timestamp and E_B uses a key Bob shares with Trent.
3. Trent generates random session key K and sends $E_A(B, R_A, K, T_B), E_B(A, K, T_B), R_B$ to Alice where E_A uses a key Alice shares with Trent.
4. Alice decrypts and confirms that R_A is her random value. She then sends to Bob $E_B(A, K, T_B), E_K(R_B)$.
5. Bob extracts K and confirms that T_B and R_B have the same value as in step 2.

Denning-Sacco

1. Alice sends A, B to Trent
2. Trent sends Alice $S_T(B, K_B), S_T(A, K_A)$
3. Alice sends Bob $E_B(S_A(K, T_A)), S_T(B, K_B), S_T(A, K_A)$
4. Bob decrypts, checks signatures and timestamps

Wide-Mouth Frog protocol

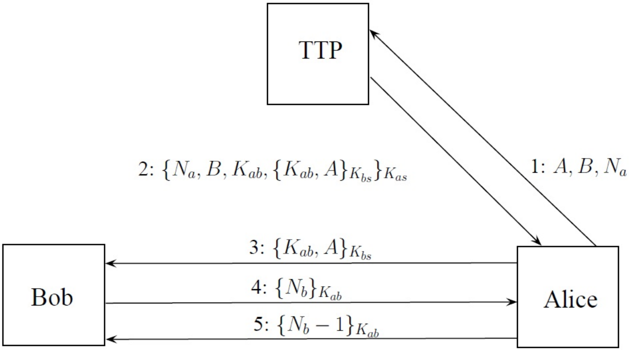


Wide-Mouth Frog protocol

The wide-mouth frog protocol has some conceptual shortcomings:

- Assumes synchronized clocks between the parties to achieve freshness.
- Although having synchronized clocks seems to be straight-forward, this is actually not the case.
  - Synchronized clocks under normal conditions is indeed easy (you have that in Windows, Linux...).
  - Synchronized clocks under attack is much harder: you need to have another protocol that securely synchronizes clocks.
  - But as soon as clock synchronization becomes security relevant, you can bet that it gets attacked.
- Bob must trust Alice that she correctly generates the session key.

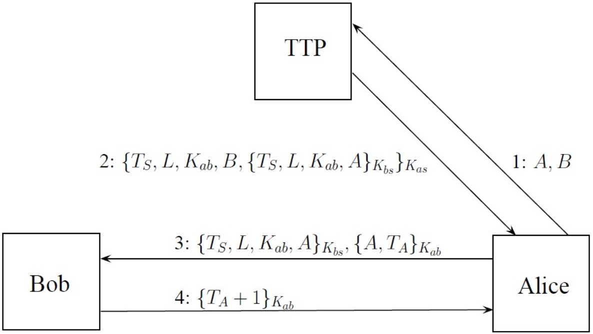
Needham-Schroeder protocol



Needham-Schroeder protocol

- Needham is one of the IT security pioneers. Protocol was conceived in 1978 and is one of the most widely studied security protocols ever.
- Removes timestamps and introduces nonces to achieve freshness.
- The session keys are generated by TTP in on the previous slide, thus removes problem of Wide-Mouth Frog protocol.
- Protocol is insecure against *known session key attacks*. Adversary who gets session key can replay the last three messages and impersonate A to B.
  - The reason for this problem is that B does not know whether the session key is fresh.
  - This vulnerability was discovered only some times after the protocol was published. Thus, even the smartest and most experienced people can fail to design secure crypto protocols.

Kerberos

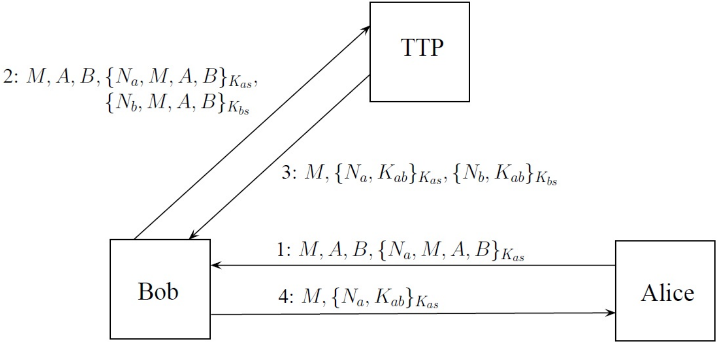


TTP
2: $\{T_S, L, K_{ab}, B, \{T_S, L, K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
1: A, B
3: $\{T_S, L, K_{ab}, A\}_{K_{bs}}, \{A, T_A\}_{K_{ab}}$
Bob
Alice
4: $\{T_A + 1\}_{K_{ab}}$

Kerberos

- Developed at MIT around 1987, made it into Windows 2000, and is still used as the authentication / key establishment / authorization mechanism within Windows.
- Quite similar to Needham-Schroeder, but removes weakness against known session key attacks using synchronized clocks.
- Shorter than Needham-Schroeder: only 4 messages instead of 5.

Otway-Rees protocol



Otway-Rees protocol

- Only 4 messages as Kerberos, but completely different messages.
- Does not require clock synchronization.
- Has a number of problems (see exercises)

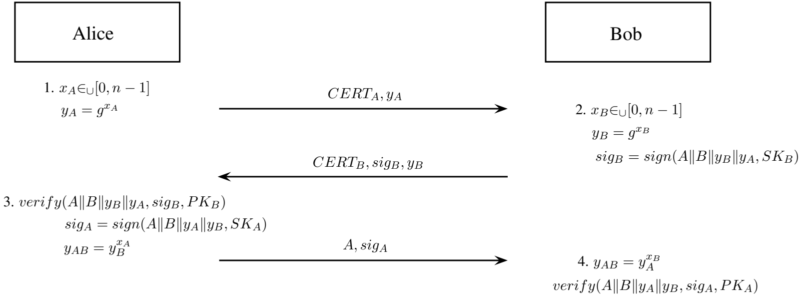
Problem

Describe possible attacks on this protocol:

1. Alice transmits $A, S_A(E_{B_{pub}}(K, R_A))$ to Bob.
2. Bob transmits $B, E_K(R_A)$ to Alice.
3. Their secure, authenticated exchange is then:
   3.1 Alice sends $E_K(i_A, M_A^{i_A}, H(i_A, M_A^{i_A}))$ to Bob.
   3.2 Bob sends $E_K(i_B, M_B^{i_B}, H(i_B, M_B^{i_B}))$ to Alice.

Station to station key agreement protocol

Common input: $\mathbb{Z}^*{}_p$ and $g \in \mathbb{Z}^*{}_p$, and n such that $g^n \equiv 1 \mod p$



- The protocol above is a simplified version of the STS protocol to illustrate the idea of authenticating messages with public keys.
- For a detailed spec refer to `http://en.wikipedia.org/wiki/Station-to-Station_protocol`

Station to station key agreement protocol

- The “station to station protocol” is the DH protocol made secure against MIM attacks:
  - The idea is simple: Alice and Bob basically sign all the messages they exchange in the Diffie - Hellman protocol.
  - The “exchange of authenticated signing keys” is done using certificates.
- Station to station protocol is the basis for the practically important *IKE* (Internet Key Exchange protocol).
- The bottom line is: one cannot establish authenticated keys without bootstrapping the system using an “exterior authentication mechanism” (e.g., without first establishing public key certificates for Alice and Bob).

RSA key transport

https://www.theinquirer.net/inquirer/news/2343117/ietf-drops-rsa-key-transport-from-ssl

Lessons Learned

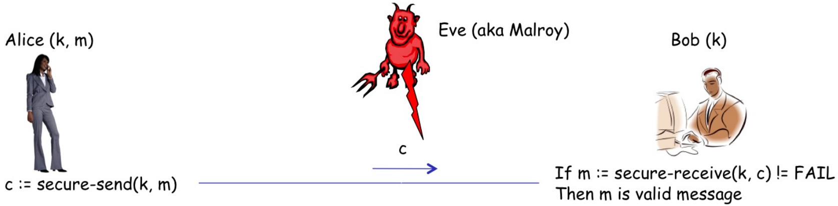
- Do not try to be too clever, over-optimization is often the cause for vulnerabilities
- Which optimizations you can do (and which optimization actually matter) depends on your assumptions (adversary model, system capabilities)
- Which protocol to use depends on your performance goals and communications capabilities (all-to-all communication, trusted party, latency, bandwidth and computational constraints)

Break

Overview

- By *secure channel* we refer to a logical channel running on top of some insecure link (typically the Internet) that provides
  - Confidentiality
  - Integrity and authenticity
  - Message freshness
- Secure channels are probably one of the most important applications of crypto in the real world.
- Many well known secure network protocols such as TLS/SSL, VPNs, IPSec, WPA etc but also application specific (e.g., secure VoIP), and proprietary protocols (maybe Skype?) make use of secure channels.
- Essentially all these protocols build upon the basic ideas we discuss in the following.
- It is also possible to get it wrong, e.g., the WEP protocol has a series of security flaws.

Secure channel



Secure channel - Secure send

```
secure-send(m, k_E, k_M) {
    STATIC msgsnt := 1
    IF (msgsnt ≥ MAX_MSGS) THEN RETURN ⊥
    c := ENC(k_E, m)
    m̃ := msgsnt||LENGTH(c)||c
    t := MAC(k_M, m̃)
    SEND(m̃||t)
    msgsnt := msgsnt + 1
}
```

Secure channel - Secure receive

secure−receive(C, k_E, k_M) {

STATIC $msgrcvd := 0$

$(msgsnt, len, c, t) = PARSE(C)$

IF $(t \neq MAC(k_M, msgsnt||len||c))$ THEN RETURN $\bot$

IF $(msgsnt \leqslant msgrcvd)$ THEN RETURN $\bot$

$m := DEC(k_E, c)$

$msgrcvd := msgsnt$

RETURN m

}

Remarks

- The *freshness property* based on counters guarantees the following: If $m_1, m_2, \ldots, m_n$ denote the messages send using secure-send(), then secure-receive() can guarantee that the messages $m_1, m_2, \ldots, m_n$ being received are subsequence of the messages sent.
- Counters give no timing guarantees, i.e., the adversary Mallory can delay messages at will.
- Timing guarantees can be achieved using
  - Time-stamps
  - Challenges
- No security protocol can prevent Mallory from discarding messages.
- MACs provide not just integrity protection but also *authenticity*, as discussed earlier.
- Further reading material: Chapter 8 in Practical Cryptography by Schneier & Ferguson.

Remarks

- Typically, secure-send() and secure-receive() are run by both parties using a secure channel.
- Each party will have an independent key-pair (enc & MAC).
- In practice, one introduces the notion of a session (e.g., e-banking). Consists of a session ID in the header, which allows the receiver to look-up session state (keys, counters etc.) when receiving a message.
- There are also other constructions for authenticated encryption, which are like block-cipher modes that guarantee at the same time confidentiality and integrity.
- For more info see, e.g., here: `http://en.wikipedia.org/wiki/Authenticated_encryption`

Forward secrecy

What happens if your private key is compromised
to your *past* communication data?

Repudiation vs. non-repudiation

- Digital signatures allow *proving* that someone said something
- Alice may be happy to authenticate to Bob, but not to Eve or Mallory!

Repudiation vs. non-repudiation

- Digital signatures allow *proving* that someone said something
- Alice may be happy to authenticate to Bob, but not to Eve or Mallory!
- Bob may turn "evil" and use Alice's statements against her later

⇒ Signatures may provide too much (authentication *and* non-repudiation)

Off-the-record (OTR) protocols allow *repudiation*

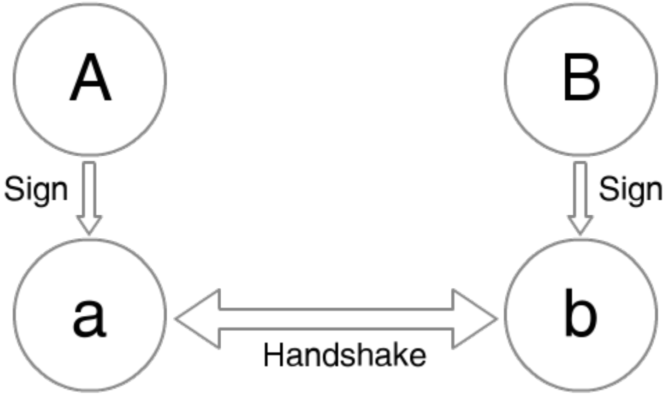
Static keys vs. ephemeral keys

Diffie-Hellman with:

- static keys allow authenticated encryption without signatures
- ephemeral keys protect against replay attacks and provide forward secrecy

OTR (Idea)

$$S_A(T_A) \tag{1}$$
$$S_B(T_B) \tag{2}$$
$$HKDF(DH(T_A, T_B)) \tag{3}$$



OTR (Real)

The OTR protocol protects the above KX by wrapping it inside another ephemeral key exchange:

$$K_1 := DH(T_A^1 || T_B^1) \tag{4}$$
$$E_{K_1}(S_A(T_A^2)) \tag{5}$$
$$E_{K_1}(S_B(T_B^2)) \tag{6}$$
$$K_2 := HKDF(DH(T_A^2, T_B^2)) \tag{7}$$
$$\tag{8}$$

To achieve forward secrecy, OTR keeps rolling out new keys $T_{A,B}^i$. To improve deniability, OTR publishes the old MAC keys once the conversation progresses.

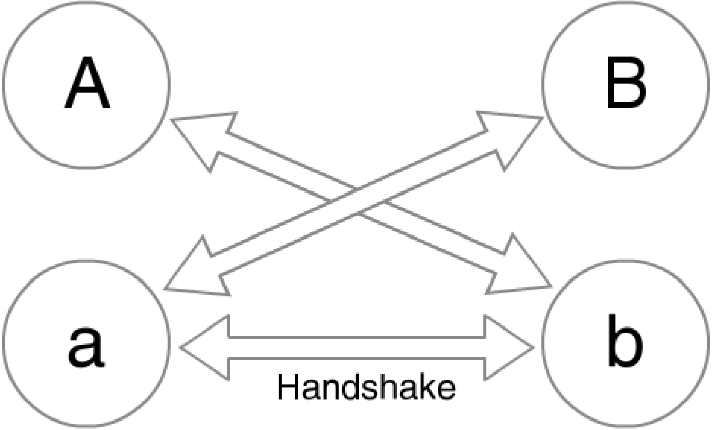
Is OTR deniable?

Is OTR deniable?

Both parties still have proof that they communicated: $S_X(T_X)$!

3DH (Trevor Perrin)

A: $K = HKDF(DH(T_a, T_B)||DH(T_a, B)||DH(a, T_B))$
B: $K = HKDF(DH(T_A, T_b)||DH(T_A, b)||DH(A, T_b))$



A Message from God (Dominic Tarr)

With 3DH, what happens if Alice's private key (a, T_a) is compromised?

A Message from God (Dominic Tarr)

With 3DH, what happens if Alice's private key (a, T_a) is compromised?

$$\text{M: } K = HKDF(DH(T_a, T_g)||DH(T_a, G)||DH(a, T_g))$$
$$\text{A: } K = HKDF(DH(T_a, T_G)||DH(T_a, G)||DH(a, T_G))$$

Asynchronous forward secrecy: SCIMP

Idea of Silence Circle's SCIMP:

Replace key with its own hash.

- New key in zero round trips!
- Forward secrecy!

Future secrecy

Suppose your regain control over your system.
What happens with your *future* communication data?

Axolotl / Signal Protocol

```
MK      CK      RK      CK      MK
--      --      --      --      --
            ECDH(A0,B0)
                |
                |
    ECDH(A1,B0) +
               /|
              / |
             /  + ECDH(A1,B1)
     CK-A1-B0   |\
         |      | \
MK-0 ----+      |  \
         |      |   CK-A1-B1
MK-1 ----+      |       |
         |      |       +---- MK-0
MK-2 ----+      |       |
                |       +---- MK-1
    ECDH(A2,B1) +
               /|
              / |
             /  |
     CK-A2-B1   |
         |      + ECDH(A2,B2)
MK-0 ----+       \
                  \
                   \
                    CK-A2-B2
```

Securing unidirectional communcation

- Alice knows Bob's public key B
- Alice wants to send M to Bob
- Alice cannot receive messages from Bob (possibly ever)

Securing unidirectional communcation

- Alice knows Bob's public key B
- Alice wants to send M to Bob
- Alice cannot receive messages from Bob (possibly ever)

Suggestion:

$$K := DH(T_A, B) \tag{9}$$

$$C := E_K(S_A(T_A, A, B)||M) \tag{10}$$

With Curve25519, cryptography has 92–128 bytes overhead:

- one or two 32 byte public keys
- one 64 byte EdDSA signature
- (plus HMAC)

What are the security properties we get here?

Break

Motivation

Suppose Alice and Bob communicate using encryption.

What can Eve still learn here?

Motivation

Suppose Alice and Bob communicate using encryption.

What can Eve still learn here?

Eve cannot read the data Alice and Bob are sending, but:

- Eve knows that Alice and Bob are communicating.
- Eve knows the amount of data they are sending and can observe patterns.

$\Rightarrow$ Patterns may even allow Eve to figure out the data

How Much does TLS leak?

"We present a traffic analysis attack against over 6000 webpages spanning the HTTPS deployments of 10 widely used, industry-leading websites in areas such as healthcare, finance, legal services and streaming video. Our attack **identifies individual pages** in the same website with 89% accuracy, exposing personal details including **medical conditions**, financial and **legal affairs** and **sexual orientation**. We examine evaluation methodology and reveal accuracy variations as large as 18% caused by assumptions affecting caching and cookies." [**?**]

https://www.youtube.com/watch?v=V2rVYvylvZc (5'2014)

Anonymity Definitions

Merriam-Webster:

1. not named or identified: "an anonymous author", "they wish to remain anonymous"
2. of unknown authorship or origin: "an anonymous tip"
3. lacking individuality, distinction, or recognizability: "the anonymous faces in the crowd", "the gray anonymous streets" – William Styron

Anonymity Definitions

Andreas Pfitzmann et. al.:

> "Anonymity is the state of being not identifiable within a set of subjects, the anonymity set."

Anonymity Definitions

Andreas Pfitzmann et. al.:

> "Anonymity is the state of being not identifiable within a set of subjects, the anonymity set."

EFF:

> "Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all)."

Anonymity Definitions

Andreas Pfitzmann et. al.:

"Anonymity is the state of being not identifiable within a set of subjects, the anonymity set."

EFF:

"Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all)."

Mine:

A user's action is anonymous if the adversary cannot link the action to the user's identity

The user's identity

includes personally identifiable information, such as:

- real name
- fingerprint
- passport number
- IP address
- MAC address
- login name
- ...

Actions

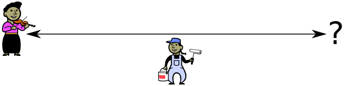
include:

- Internet access
- speach
- participation in demonstration
- purchase in a store
- walking across the street
- ...

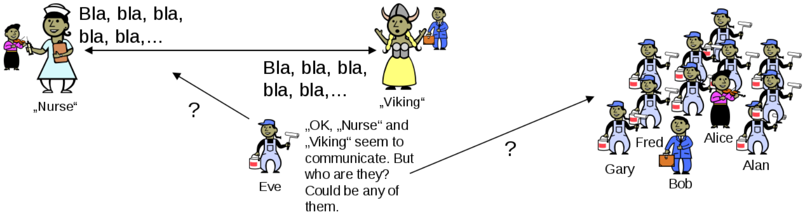
Anonymity: Terminology

- Sender Anonymity: The initiator of a message is anonymous. However, there may be a path back to the initiator.

- Receiver Anonymity: The receiver of a message is anonymous.



Pseudonymity



Pseudonymity

- A pseudonym is an identity for an entity in the system. It is a "false identity" and not the true identity of the holder of the pseudonym.
- Nobody, but (maybe) a trusted party may be able to link a pseudonym to the true identity of the holder of the pseudonym.
- A pseudonym can be tracked. We can observe its behaviour, but we do not learn who it is.

Evaluating Anonymity

How much anonymity does a given system provide?

- Number of known attacks?
- Lowest complexity of successful attacks?
- Information leaked through messages and maintenance procedures?
- Number of users?

Anonymity: Basics

- **Anonymity Set** is the set of suspects
- Attacker computes a **probability distribution** describing the likelyhood of each participant to be the responsible party.
- Anonymity is the stronger, the larger the anonymity set and the more evenly distributed the subjects within that set are.

Anonymity Metric: Anonymity Set Size

Let $\mathcal{U}$ be the attacker's probability distribution and $p_u = \mathcal{U}(u)$ describing the probability that user $u \in \Psi$ is responsible.

$$ASS := \sum_{\substack{u \in \Psi \\ p_u > 0}} 1 \qquad (11)$$

Large Anonymity Sets

Examples of large anonymity sets:

- Any human

Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human with Internet access

Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human with Internet access
- Any human speaking German

Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human with Internet access
- Any human speaking German
- Any human speaking German with Internet access awake at 3am CEST

Anonymity Metric: Maximum Likelihood

Let $\mathcal{U}$ be the attacker's probability distribution describing the probability that user $u \in \Psi$ is responsible.

$$ML := \max_{u \in \Psi} p_u \qquad (12)$$

Anonymity Metric: Maximum Likelihood

- For successful criminal prosecution in the US, the law requires ML *close* to 1 ("beyond reasonable doubt")
- For successful civil prosecution in the US, the law requires $ML > \frac{1}{2}$ ("more likely than not")
- For a given anonymity set, the best anonymity is achieved if

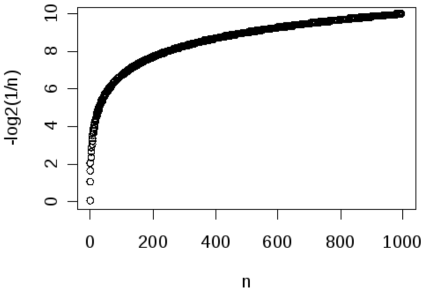
$$ML = \frac{1}{ASS} \tag{13}$$

Anonymity Metric: Entropy

Let $\mathcal{U}$ be the attacker's probability distribution describing the probability that user $u \in \Psi$ is responsible. Define the effective size S of the anonymity distribution $\mathcal{U}$ to be:

$$S := -\sum_{u \in \Psi} p_u \log_2 p_u \quad (14)$$

where $p_u = \mathcal{U}(u)$.



Interpretation of Entropy

$$S = -\sum_{u \in \Psi} p_u \log_2 p_u \tag{15}$$

This is the *expected* number of bits of additional information that the attacker needs to definitely identify the user (with absolute certainty).

Entropy Calculation Example

Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is S?

Entropy Calculation Example

Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is S?

- For 101 nodes $H_{max} = 6.7$
-

$$S = -\frac{100 \cdot \log_2 0.001}{1000} - \frac{9 \cdot \log_2 0.9}{10} \tag{16}$$

$$\approx 0.9965 + 0.1368 \tag{17}$$

$$= 1.133... \tag{18}$$

Attacks to avoid

Hopeless situations include:

- All nodes collaborate against the victim
- All directly adjacent nodes collaborate
- All non-collaborating adjacent nodes are made unreachable from the victim
- The victim is required to prove his innocence

Economics & Anonymity

R. Dingledine and P. Syverson wrote about *Open Issues in the Economics of Anonymity*:

- Providing anonymity services has economic disincentives (DoS, legal liability)
- Anonymity requires introducing inefficiencies

⇒ Who pays for that?

Economics & Anonymity

R. Dingledine and P. Syverson wrote about *Open Issues in the Economics of Anonymity*:

- Providing anonymity services has economic disincentives (DoS, legal liability)
- Anonymity requires introducing inefficiencies

$\Rightarrow$ Who pays for that?

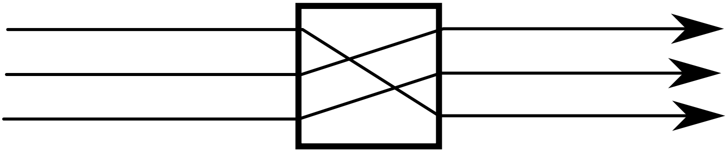
The anonymizing server that has the best reputation (performance, most traffic) is presumably compromised.

Anonymity: Dining Cryptographers

"Three cryptographers are sitting down to dinner. The waiter informs them that the bill will be paid anonymously. One of the cryptographers maybe paying for dinner, or it might be the NSA. The three cryptographers respect each other's right to make an anonymous payment, but they wonder if the NSA is paying." – David Chaum

Mixing

David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:

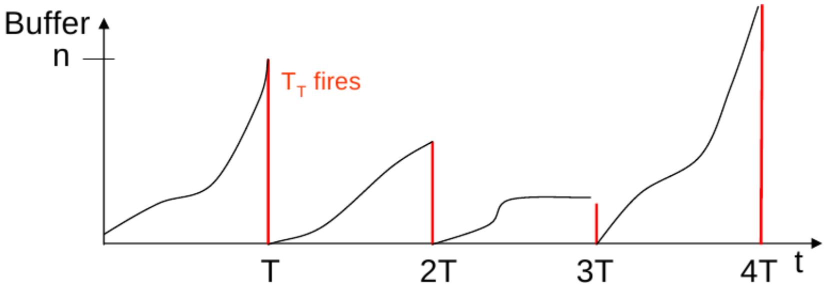


Mixing

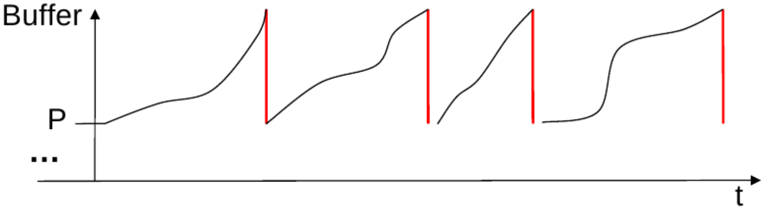
David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:

Threshold Mix

Timed Mix



Pool mix



Break

Tor

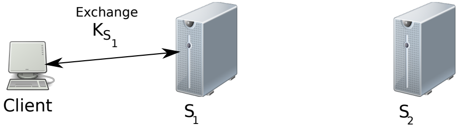
- Tor is a P2P network of **low-latency** mixes which are used to provide anonymous communication between parties on the Internet.
- Tor works for any TCP-based protocol
- TCP traffic enters the Tor network via a SOCKS proxy
- **Common usage:** client anonymity for web browsing

Onion Routing

- Multiple mix servers
- Path of mix servers chosen by initiator
- Chosen mix servers create "circuit"
  - Initiator contacts first server S_1, sets up symmetric key K_{S_1}
  - Then asks first server to connect to second server S_2; through this connection sets up symmetric key with second server K_{S_2}
  - ...
  - Repeat with server S_i until circuit of desired length n constructed

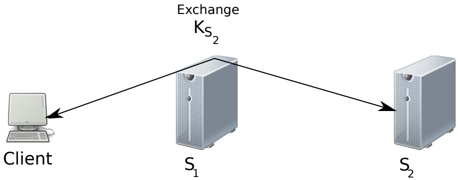
Onion Routing Example

- Client sets up symmetric key K_{S_1} with server S_1



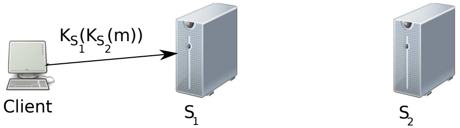
Onion Routing Example

- Via S_1 Client sets up symmetric key K_{S_2} with server S_2



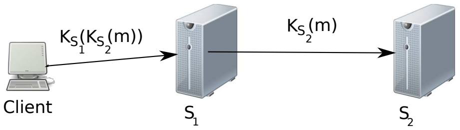
Onion Routing Example

- Client encrypts m as $K_{S_1}(K_{S_2}(m))$ and sends to S_1



Onion Routing Example

- S_1 decrypts, sends on to S_2, S_2 decrypts, revealing m

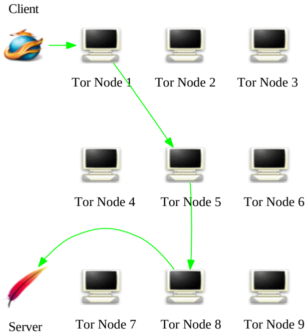


Tor - How it Works

- Low latency P2P Network of mix servers
- Designed for interactive traffic (https, ssh, etc.)
- "Directory Servers" store list of participating servers
  - Contact information, public keys, statistics
  - Directory servers are replicated for security
- Clients choose servers randomly with bias towards high BW/uptime
- Clients build long lived Onion routes "circuits" using these servers
- Circuits are bi-directional
- Circuits are of length three

Tor - How it Works - Example

- Example of Tor client circuit



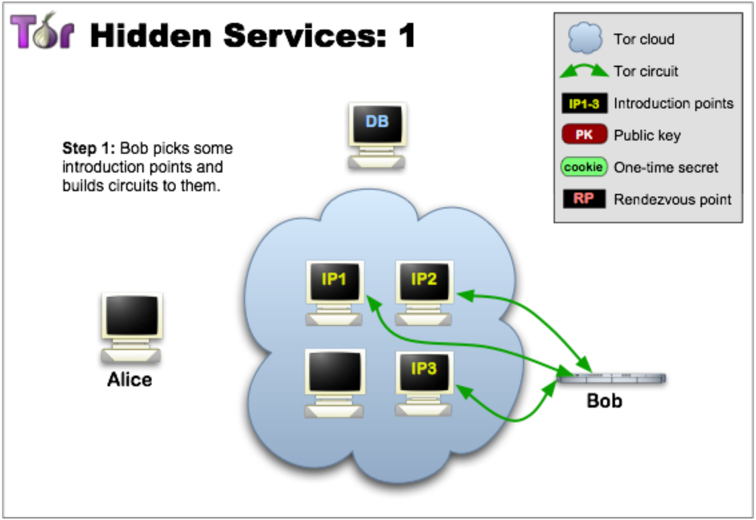
Tor - How it Works - Servers

- Servers are classified into three categories for usability, security and operator preference
- Entry nodes (aka guards) - chosen for first hop in circuit
  - Generally long lived "good" nodes
  - Small set chosen by client which are used for client lifetime (security)
- Middle nodes - chosen for second hop in circuit, least restricted set
- Exit nodes - last hop in circuit
  - Visible to outside destination
  - Support filtering of outgoing traffic
  - Most vulerable position of nodes

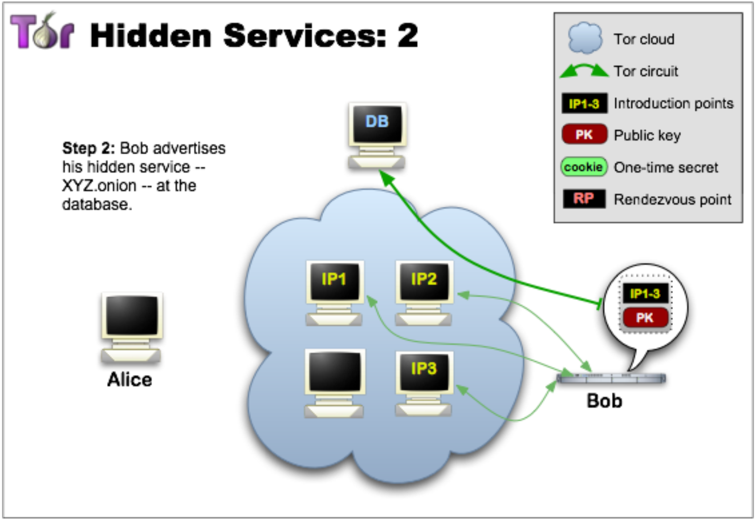
Hidden Services in Tor

- Hidden services allow Tor servers to receive incoming connections anonymously
- Can provide access to services available *only* via Tor
  - Web, IRC, etc.
  - For example, host a website without your ISP knowing

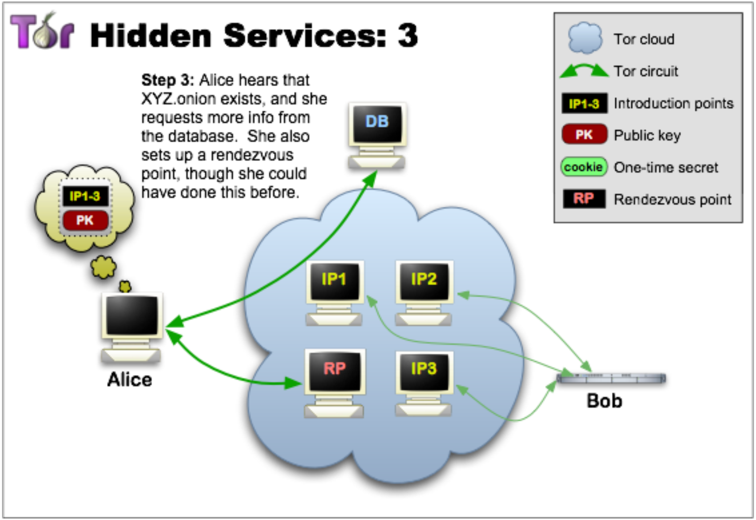
Hidden Services Example 1



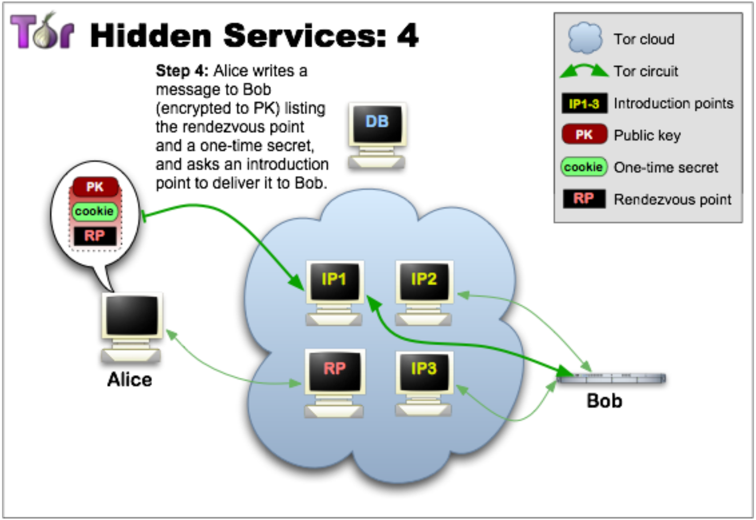
Hidden Services Example 2



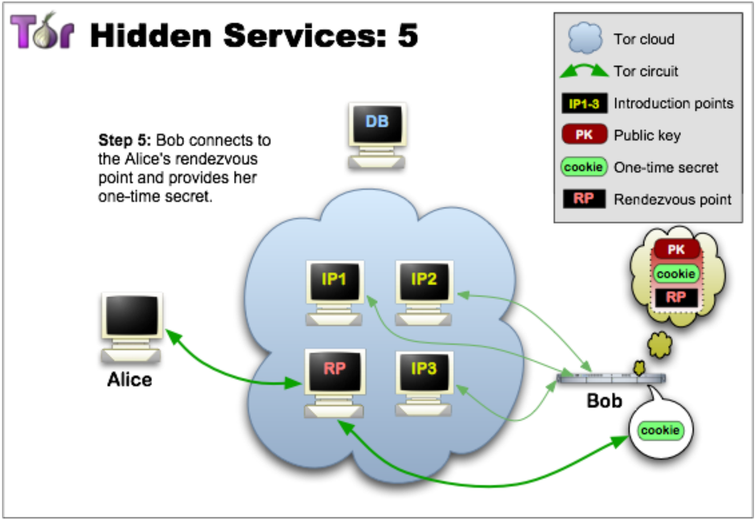
Hidden Services Example 3



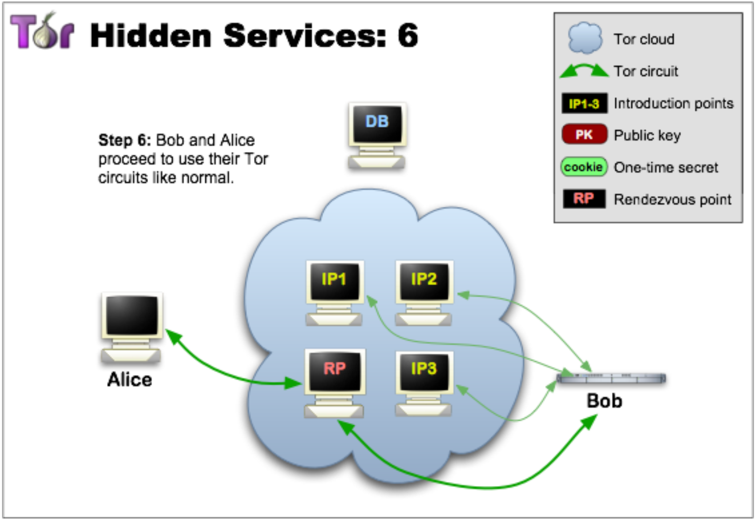
Hidden Services Example 4



Hidden Services Example 5



Hidden Services Example 6



Types of Attacks on Tor

- Exit Relay Snooping
- Website fingerprinting
- Traffic Analysis
- Intersection Attack
- DoS

Exercise

- Install Tor
- Configure Tor relay
- Setup hidden service
- Perform risk analysis for deanonymization

Acknowlegements

This presentation used material from:

- https://signal.org/blog/simplifying-otr-deniability/
- Endre Bangerter (BTI 7261/2017)

References