Cryptography in GNUnet

Protocols for a Future Internet for Libre Societies

Christian Grothoff

Sept 30th, 2015

Sometime in 2013...

The ~~NEW~~GNU Network (very simplified)

Internet

Google
DNS/X.509
TCP/UDP
IP/BGP
Ethernet
Phys. Layer

The ~~NEW~~GNU Network (very simplified)

Internet	
Google	
DNS/X.509	
TCP/UDP	
IP/BGP	
Ethernet	
Phys. Layer	HTTPS/TCP/WLAN/...

The ~~NEW~~GNU Network (very simplified)

Internet

Google
DNS/X.509
TCP/UDP
IP/BGP
Ethernet
Phys. Layer

CORE
HTTPS/TCP/WLAN/...

The ~~NEW~~GNU Network (very simplified)

Internet	
Google	
DNS/X.509	
TCP/UDP	
IP/BGP	R^5N DHT
Ethernet	CORE
Phys. Layer	HTTPS/TCP/WLAN/...

The ~~NEW~~GNU Network (very simplified)

Internet	
Google	
DNS/X.509	
TCP/UDP	CADET
IP/BGP	R^5N DHT
Ethernet	CORE
Phys. Layer	HTTPS/TCP/WLAN/...

The ~~NEW~~GNU Network (very simplified)

Internet	
Google	
DNS/X.509	GNU Name System
TCP/UDP	CADET
IP/BGP	R^5N DHT
Ethernet	CORE
Phys. Layer	HTTPS/TCP/WLAN/...

The ~~NEW~~GNU Network (very simplified)

Internet
Google
DNS/X.509
TCP/UDP
IP/BGP
Ethernet
Phys. Layer

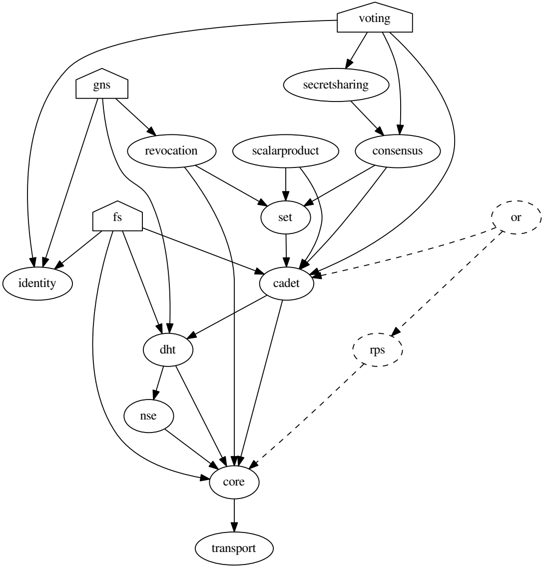
Applications
GNU Name System
CADET
R^5N DHT
CORE
HTTPS/TCP/WLAN/...

The ~~NEW~~GNU Network (very simplified)

Internet
Google
DNS/X.509
TCP/UDP
IP/BGP
Ethernet
Phys. Layer

GNUnet
Applications
GNU Name System
CADET
R^5N DHT
CORE
HTTPS/TCP/WLAN/...

The ~~NEW~~GNU Network (still simplified)



Chapter 1: Public Key Infrastructure

Chapter 1: Public Key Infrastructure

Remark: Public Keys $\not\subseteq$ Public Information

Censorship-Resistant Sharing

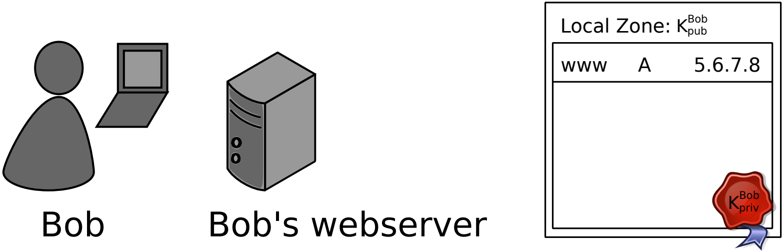
Design objectives

- **Authorized** users can decrypt shared data
- **Intermediaries** can verify reply matches request
- **Intermediaries** cannot decrypt shared data
- **Intermediaries** cannot understand query, other than via guessing / confirmation attack
- Cost of all operations is $O(1)$, bandwidth overheads $< 100/\mathit{bytes}$ per request

Consequences

- P2P overlay can be used to efficiently **replicate** or **cache** data (impossible with end-to-end encryption)
- Peers in the overlay cannot effectively **censor** or efficiently **spy** on participants

Name resolution in the GNU Name System

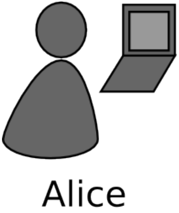


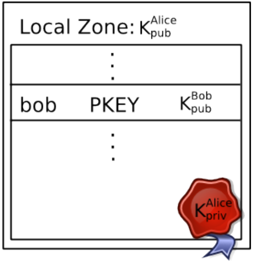
- Bob can locally reach his webserver via **www.gnu**

Secure introduction

- Bob gives his public key to his **friends**, possibly via QR code

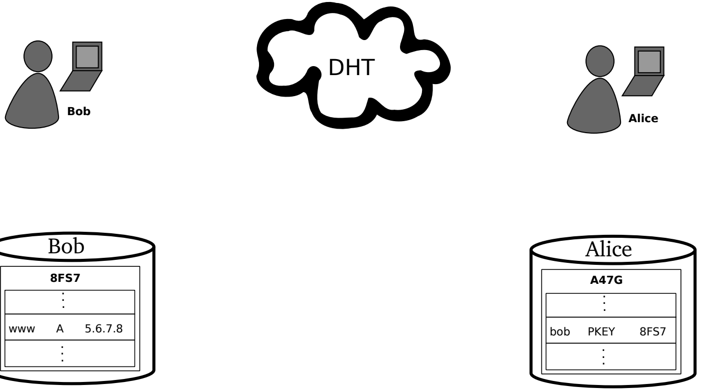
Delegation



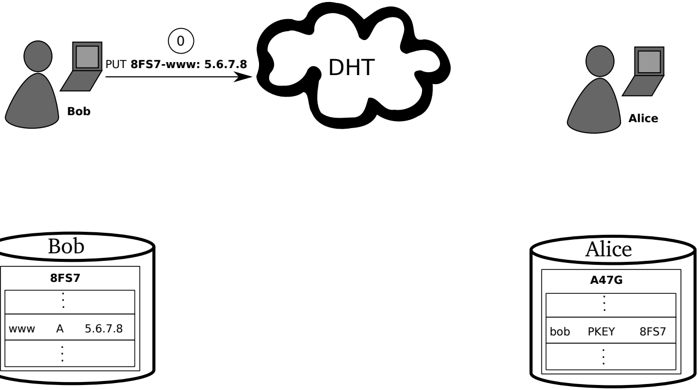


- Alice learns Bob's public key
- Alice creates delegation to zone K_{pub}^{Bob} under label **bob**
- Alice can reach Bob's webserver via **www.bob.gnu**

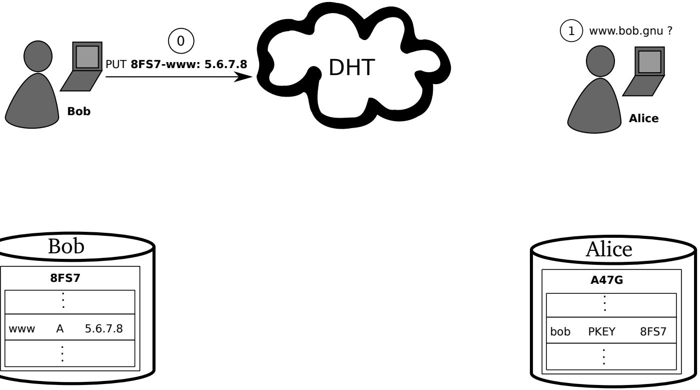
Name Resolution



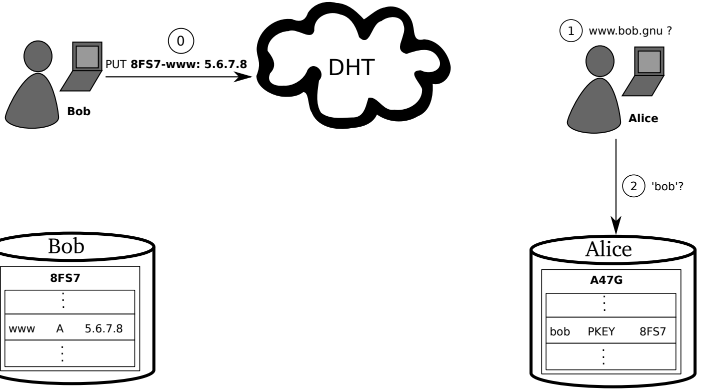
Name Resolution



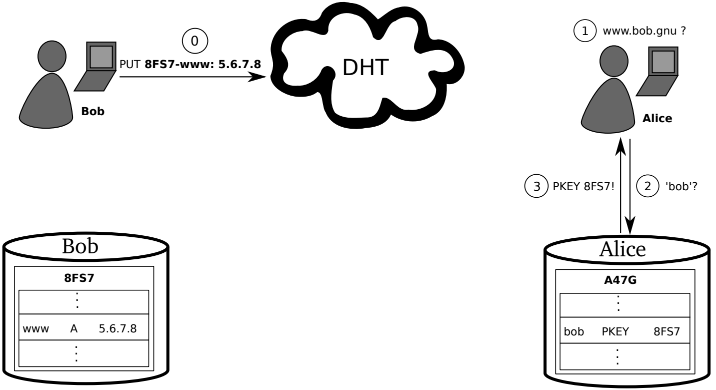
Name Resolution



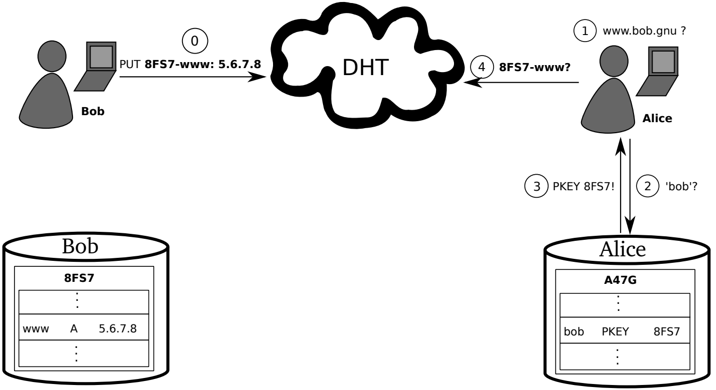
Name Resolution



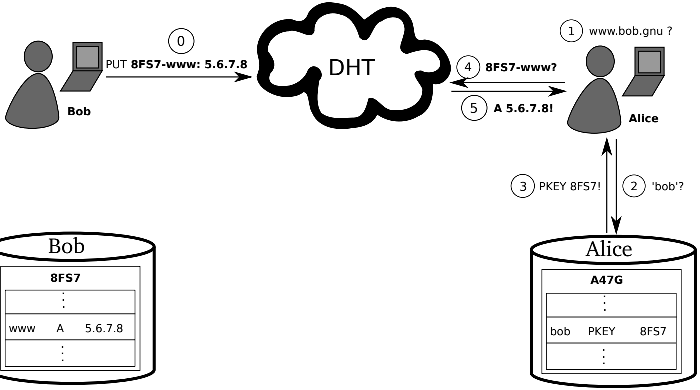
Name Resolution



Name Resolution



Name Resolution



Query Privacy: Terminology

G	generator in ECC curve, a point
n	size of ECC group, $n := \|G\|$, n prime
x	private ECC key of zone ($x \in \mathbb{Z}_n$)
P	public key of zone, a point $P := xG$
l	label for record in a zone ($l \in \mathbb{Z}_n$)
$R_{P,l}$	set of records for label l in zone P
$q_{P,l}$	query hash (hash code for DHT lookup)
$B_{P,l}$	block with encrypted information for label l in zone P published in the DHT under $q_{P,l}$

Query Privacy: Cryptography

Publishing records $R_{P,l}$ as $B_{P,l}$ under key $q_{P,l}$

$$h := H(l, P) \tag{1}$$

$$d := h \cdot x \mod n \tag{2}$$

$$B_{P,l} := S_d(E_{HKDF(l,P)}(R_{P,l})), dG \tag{3}$$

$$q_{P,l} := H(dG) \tag{4}$$

Query Privacy: Cryptography

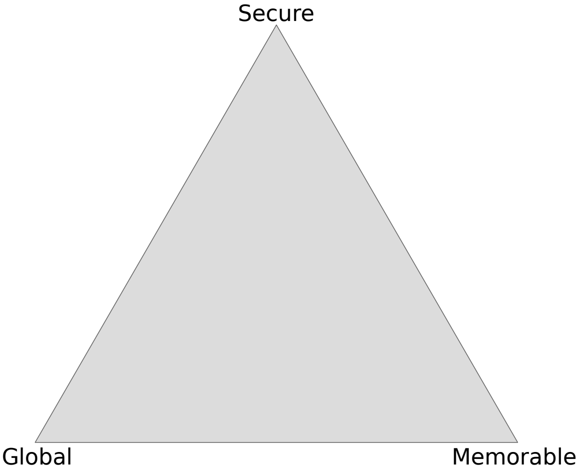
Publishing records $R_{P,l}$ as $B_{P,l}$ under key $q_{P,l}$

$$
\begin{aligned}
h &:= H(l, P) && (1)\\
d &:= h \cdot x \mod n && (2)\\
B_{P,l} &:= S_d(E_{HKDF(l,P)}(R_{P,l})), dG && (3)\\
q_{P,l} &:= H(dG) && (4)
\end{aligned}
$$

Searching for records under label l in zone P

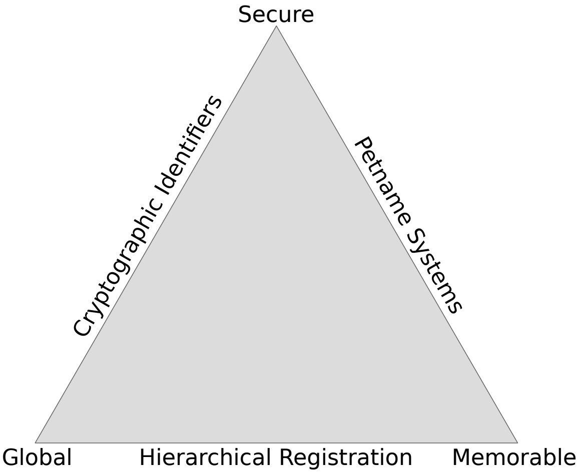
$$
\begin{aligned}
h &:= H(l, P) && (5)\\
q_{P,l} &:= H(hP) = H(hxG) = H(dG) \Rightarrow \texttt{obtain}\ B_{P,l} && (6)\\
R_{P,l} &= D_{HKDF(l,P)}(B_{P,l}) && (7)
\end{aligned}
$$

Zooko's Triangle



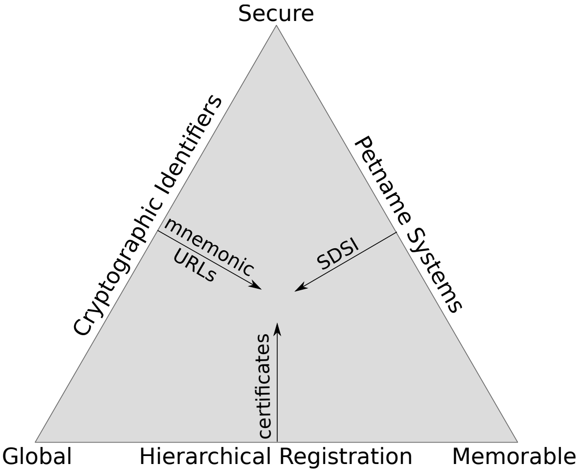
A name system can only fulfill **two**!

Zooko's Triangle



DNS, ".onion" IDs and `/etc/hosts/` are representative designs.

Zooko's Triangle



DNSSEC security is broken by design (adversary model!)

Summary: The GNU Name System[1]

Properties of GNS

- Decentralized name system with secure memorable names
- Delegation used to achieve transitivity
- Supports globally unique, secure identifiers
- Achieves query and response privacy
- Provides alternative public key infrastructure
- Interoperable with DNS

New applications enabled by GNS

- Name services hosted in P2P networks
- Name users in decentralized social networking applications

[1] Joint work with Martin Schanzenbach and Matthias Wachs

Chapter 2: Privacy-preserving Computation

Scalarproduct for GNUnet[2]

Motivation

- Scalarproduct trivially provides cosine similarity
- Useful for information retrieval and data mining
- Our envisioned application:

  privacy-preserving collaborative ranking in news distribution

Properties

- Scalarproduct over map on intersecting sets, not just vectors
- Privacy-preserving (but need to limit number of interactions)
- Relatively efficient in bandwidth and CPU usage

---

[2] Joint work with Tanja Lange and Christian Fuchs

Background: Paillier

We use the Paillier cryptosystem:

$$\mathrm{E}_K(m) := g^m \cdot r^n \mod n^2, \tag{8}$$

$$\mathrm{D}_K(c) := \frac{(c^\lambda \mod n^2) - 1}{n} \cdot \mu \mod n \tag{9}$$

where the public key $K = (n, g)$, m is the plaintext, c the ciphertext, n the product of $p, q \in \mathbb{P}$ of equal length, and $g \in \mathbb{Z}^*_{n^2}$. The private key is (λ, μ), which is computed from p and q as follows:

$$\lambda := \mathrm{lcm}(p-1, q-1), \tag{10}$$

$$\mu := \left(\frac{(g^\lambda \mod n^2) - 1}{n}\right)^{-1} \mod n. \tag{11}$$

Paillier offers **additive** homomorphism

Paillier offers additive homomorphic public-key encryption, that is:

$$\mathsf{E}_K(a) \otimes \mathsf{E}_K(b) \equiv \mathsf{E}_K(a + b) \tag{12}$$

for some public key K.

Background: Secure Multiparty Computation

- Alice and Bob have private inputs a_i and b_i.
- Alice and Bob run a protocol to jointly calculate $f(a_i, b_i)$.
- One of them learns the result.
- Adversary model: honest but curious

Secure Scalar Product

- Original idea by Ioannids et al. in 2002 (use: $(a-b)^2 = a^2 - 2ab + b^2$)
- Refined by Amirbekyan et al. in 2007 (corrected math)
- Implemented with practical extensions in GNUnet (negative numbers, small numbers, concrete protocol, set intersection, implementation).

Preliminaries

- Alice has public key A and input map $m_A : M_A \to \mathbb{Z}$.
- Bob has public key B and input map $m_B : M_B \to \mathbb{Z}$.
- We want to calculate

$$\sum_{i \in M_A \cap M_B} m_A(i) m_B(i) \tag{13}$$

- We first calculate $M = M_A \cap M_B$.
- Define $a_i := m_A(i)$ and $b_i := m_B(i)$ for $i \in M$.
- Let s denote a shared static offset.

Network Protocol

- Alice transmits $\mathsf{E}_A(s + a_i)$ for $i \in M$ to Bob.
- Bob creates two random permutations π and π' over the elements in M, and a random vector r_i for $i \in M$ and sends

$$
\begin{aligned}
R :&= \mathsf{E}_A(s + a_{\pi(i)}) \otimes \mathsf{E}_A(s - r_{\pi(i)} - b_{\pi(i)}) && (14)\\
&= \mathsf{E}_A(2 \cdot s + a_{\pi(i)} - r_{\pi(i)} - b_{\pi(i)}), && (15)\\
R' :&= \mathsf{E}_A(s + a_{\pi'(i)}) \otimes \mathsf{E}_A(s - r_{\pi'(i)}) && (16)\\
&= \mathsf{E}_A(2 \cdot s + a_{\pi'(i)} - r_{\pi'(i)}), && (17)\\
S :&= \sum (r_i + b_i)^2, && (18)\\
S' :&= \sum r_i^2 && (19)
\end{aligned}
$$

Decryption (1/3)

Alice decrypts R and R' and computes for $i \in M$:

$$a_{\pi(i)} - b_{\pi(i)} - r_{\pi(i)} = \mathrm{D}_A(R) - 2 \cdot s, \tag{20}$$

$$a_{\pi'(i)} - r_{\pi'(i)} = \mathrm{D}_A\left(R'\right) - 2 \cdot s, \tag{21}$$

which is used to calculate

$$T := \sum_{i \in M} a_i^2 \tag{22}$$

$$U := -\sum_{i \in M} (a_{\pi(i)} - b_{\pi(i)} - r_{\pi(i)})^2 \tag{23}$$

$$U' := -\sum_{i \in M} (a_{\pi'(i)} - r_{\pi'(i)})^2 \tag{24}$$

Decryption (2/3)

She then computes

$$
\begin{aligned}
P :&= S + T + U \\
&= \sum_{i \in M} (b_i + r_i)^2 + \sum_{i \in M} a_i^2 + \left(- \sum_{i \in M} (a_i - b_i - r_i)^2 \right) \\
&= \sum_{i \in M} \left((b_i + r_i)^2 + a_i^2 - (a_i - b_i - r_i)^2 \right) \\
&= 2 \cdot \sum_{i \in M} a_i (b_i + r_i). \\
P' :&= S' + T + U' \\
&= \sum_{i \in M} r_i^2 + \sum_{i \in M} a_i^2 + \left(- \sum_{i \in M} (a_i - r_i)^2 \right) \\
&= \sum_{i \in M} \left(r_i^2 + a_i^2 - (a_i - r_i)^2 \right) = 2 \cdot \sum_{i \in M} a_i r_i.
\end{aligned}
$$

Decryption (3/3)

Finally, Alice computes the scalar product using:

$$\frac{P - P'}{2} = \sum_{i \in M} a_i(b_i + r_i) - \sum_{i \in M} a_i r_i = \sum_{i \in M} a_i b_i. \tag{25}$$

Performance Evaluation[3]

Length	RSA-2048	RSA-1024
25	14 s	3 s
50	21 s	5 s
100	39 s	7 s
200	77 s	13 s
400	149 s	23 s
800	304 s	32 s

[3] Wall-clock, loopback, single-core i7 920 at 2.67 GHz

Secure Scalar Product: ElGamal/ECC-Variant

Alice's public key is $A = g^a$, her private key is a. Alices sends to Bob $(g_i, h_i) = (g^{r_i}, g^{r_i a + a_i})$ using random values r_i for $i \in M$.
Bob responds with

$$\left(\prod_{i\in M} g_i^{b_i}, \prod_{i\in M} h_i^{b_i}\right) = \left(\prod_{i\in M} g_i^{b_i}, (\prod_{i\in M} g_i^{b_i})^a g^{\sum_{i\in M} a_i b_i}\right)$$

Alice can then compute

$$\left(\prod_{i\in M} g_i^{b_i}\right)^{-a} \cdot \left(\prod_{i\in M} g_i^{b_i}\right)^{a} \cdot g^{\sum_{i\in M} a_i b_i} = g^{\sum_{i\in M} a_i b_i}.$$

Assuming $\sum_{i\in M} a_i b_i$ is sufficiently small, Alice can then obtain the scalar product by solving the DLP.

Performance Evaluation

Length	RSA-2048	ECC-2^{20}	ECC-2^{28}
25	14 s	2 s	29 s
50	21 s	2 s	29 s
100	39 s	2 s	29 s
200	77 s	3 s	30 s
400	149 s	OOR	31 s
800	304 s	OOR	33 s
800	3846 kb	OOR	70 kb

The pre-calculation of ECC-2^{28} is $\times 16$ more expensive than for ECC-2^{20} as the table is set to have size $\sqrt{n}$.

Scalarproduct: Summary

- Homomorphic encryption probably fast enough for real applications
- ECC/DLP-variant significantly better for small products or with cost amortization over multiple runs
- Future privacy-enhancing applications should consider secure **communication** and secure **computation**

Chapter 3: Electronic Cash

GNU Taler

Modern economies need a currency.

Motivation

Modern economies need a currency online.

SWIFT?

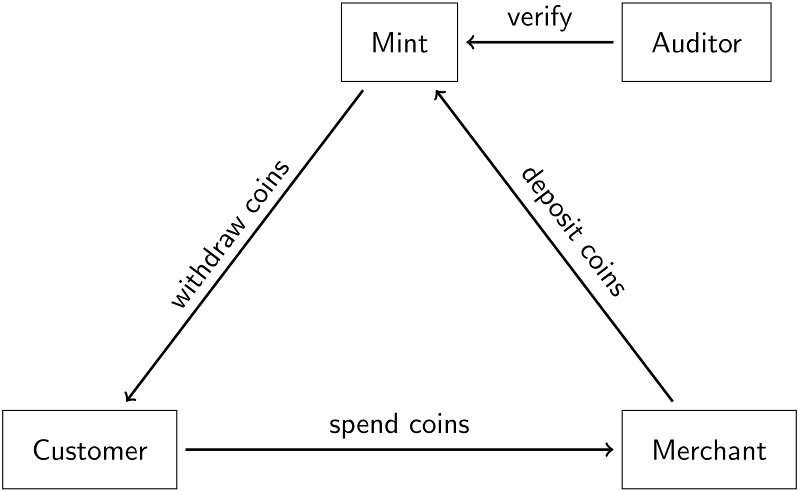
SWIFT/Mastercard/Visa are too transparent.

Let's make cash **digital** and **socially responsible**.

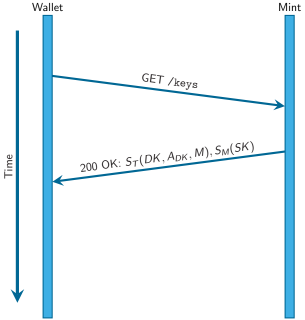
Let's make cash **digital** and **socially responsible**.

Taxable, Anonymous, Libre, Practical, Resource Friendly

Architecture of GNU Taler

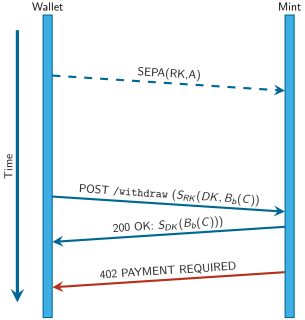


Taler `/keys`



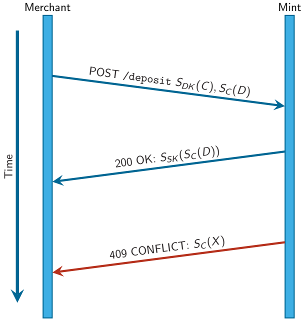
T	Financial regulator key
DK	RSA public key ("denomination key")
A_{DK}	Value of coins signed by DK
M	Offline master key of mint
SK	Online signing key of mint

Taler /withdraw/sign



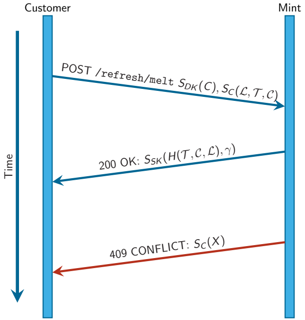
RK	Reserve key
A	Some amount, $A \geq A_{DK}$
b	Blinding factor
$B_b()$	RSA blinding
C	Coin key
$S_{DK}()$	(Blind) signature

Taler /deposit



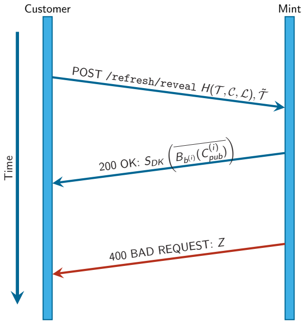
DK	Denomination key
$S_{DK}()$	RSA signature using DK
C	Coin key
$S_C()$	EdDSA signature using C
D	Deposit details
SK	Signing key
$S_{SK}()$	EdDSA signature using SK
X	Conficting deposit details

Taler /refresh/melt



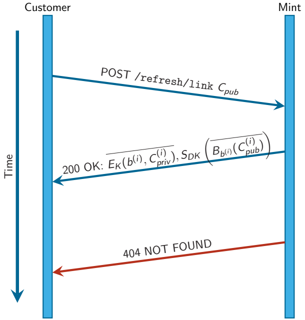
κ	System-wide security parameter
K	$:= ECDHE(T, C)$
$E_K()$	Symmetric encryption using key K
$DK^{(i)}$	List of denomination keys
$C^{(i)}$	List of coin keys
$b^{(i)}$	List of blinding factors
$B_{b^{(i)}}()$	Blinding with respective $b^{(i)}$
$\mathcal{T}$	$[T_{pub}]_\kappa$
$\mathcal{L}$	$[E_K(b^{(i)}, C^{(i)}_{priv})]_\kappa$
$\mathcal{C}$	$[B_{b^{(i)}}(C^{(i)}_{pub}), DK^{(i)}]_\kappa$
γ	Random value in $[0, \kappa)$

Taler /refresh/reveal



$\tilde{\mathcal{T}}$	$[T_{priv}]_{\kappa \setminus \gamma}$
$\overline{B_{b(i)}(C^{(i)})}$	Blinded coins from $\mathcal{C}$ at γ
Z	Cut-and-choose missmatch information

Taler /refresh/link



$\overline{E_K(b^{(i)}, C^{(i)}_{priv})}$ Linkage data $\mathcal{L}$ at γ

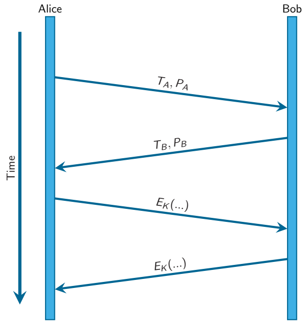
GNU Taler: Summary

Taler compared to Chaum's DigiCash

- – Only online transactions (Chaum supported off-line)
- ◦ All income based on Taler transactions visible to the state
- ◦ Supports anonymous payments
- \+ Supports spending fractions of a coin (giving change)
- \+ Change can be made unlinkable to original transaction
- \+ Can support refunds to anonymous customers
- \+ Supports microdonations (borrowing ideas from Peppercoin)
- \+ Modern, RESTful API (with modernizations in primitives)
- \+ Free software, open protocol, no patents

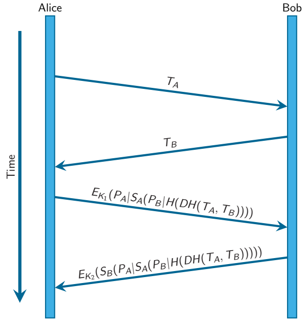
Chapter 4: Key Exchange

3DH (trevp?)



P_A	Public EdDSA key of Alice
P_B	Public EdDSA key of Bob
T_A	Ephemeral key from Alice
T_B	Ephemeral key from Bob
K	Key derived from $DH(T_A, T_B) \| DH(T_A, P_B) \| DH(P_A, T_B)$

Fixing the Wildcard (Tarr)[4]



P_A	Public EdDSA key of Alice
P_B	Public EdDSA key of Bob
T_A	Ephemeral key from Alice
T_B	Ephemeral key from Bob
K_1	Key derived from $DH(T_A, T_B)\|DH(T_A, P_B)$
K_2	Key derived from $DH(T_A, T_B)\|DH(T_A, P_B)\|DH(P_A, T_B)$

[4] http://dominictarr.github.io/secret-handshake-paper/shs.pdf

Deniable signatures (Burdges, Grothoff)

Assume $Q_a = d_A G$ and $z = H(m)$. As in ECDSA, pick random $k \in [1, n-1]$. Let $C := C_A + C_B$ be the random offset.

$$(x_1, y_1) := kG \qquad \underline{+C} \tag{26}$$

$$r := x_1 \mod n \tag{27}$$

$$s := k^{-1}(z + rd_A) \mod n \tag{28}$$

Repeat until $r, s \neq 0$. To verify:

$$w := s^{-1} \mod n \tag{29}$$

$$u_1 := zw \mod n \tag{30}$$

$$u_2 := rw \mod n \tag{31}$$

$$(x_1, y_1) := u_1 G + u_2 Q_A \qquad \underline{+C} \tag{32}$$

$$r \equiv x_1 \mod n? \tag{33}$$

Falsification of a deniable signature

Assume $Q_a = d_A G$ and $z = H(m)$. As in ECDSA, pick random $r, s, k \in [1, n-1]$. Bob does not know d_A. So he calculates:

$$w := s^{-1} \mod n \tag{34}$$

$$u_1 := zw \mod n \tag{35}$$

$$u_2 := rw \mod n \tag{36}$$

$$(x_1, y_1) := u_1 G + u_2 Q_A \tag{37}$$

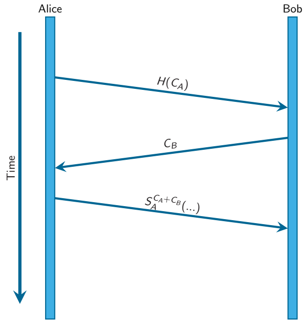
$$C \equiv x_1 - r \mod n \tag{38}$$

Bob now picks a random C_A and sets

$$C_B = C - C_A. \tag{39}$$

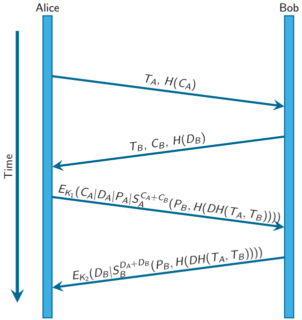
For this C_A, C_B the "random" values (r, s) are a valid signature (per construction).

Deniable signatures illustrated



- C_A Randomly chosen offset from Alice
- C_B Randomly chosen offset from Bob
- S_A^C Deniable signature using offset C and private key A

Burdges, Grothoff + Tarr



P_A	Public EdDSA key of Alice
P_B	Public EdDSA key of Bob
C_A	Randomly chosen offset from Alice
C_B	Randomly chosen offset from Bob
D_A	Randomly chosen offset from Alice
D_B	Randomly chosen offset from Bob
T_A	Ephemeral key from Alice
T_B	Ephemeral key from Bob
K_1	Key derived from $DH(T_A, T_B)\|DH(T_A, P_B)$
K_2	Key derived from $DH(T_A, T_B)\|DH(T_A, P_B)\|DH(P_A, T_B)$

KX Evolution

1. DH, STS, TLS, SSH (does sign, not deniable, no wildcard)
2. CurveCP, OTR, TextSecure, Axolotl (do not sign, deniable, wildcard)
3. Tarr (does sign, not deniable, no wildcard, expensive)
4. BG+T (fully deniable, no wildcard, still expensive)

More Information

- Florian Dold on the Cramer-style electronic voting protocol implemented in GNUnet: `https://gnunet.org/31c3videos`
- Nicolas Benes on hardware-based intrusion detection for your home router: `https://gnunet.org/31c3videos`
- Julian Kirsch on defeating port scanners: `https://gnunet.org/ghm2014knock`
- Markus Teich on data minimization for bug reporting: `https://gnunet.org/markus2013bsdefense`
- Christian Grothoff and Florian Dold on GNS and revocation in GNUnet: `https://gnunet.org/video-30c3-talk-gnu-name-system`

Conclusion

- Decentralization is necessary
- Decentralization creates challenges for research:
  - Privacy-enhancing network protocol design
  - Secure software implementations
  - Software engineering and system architecture

Questions?

Find more information at:

- https://gnunet.org/
- https://gnunet.org/videos
- http://www.taler.net/

Slides will be at http://grothoff.org/christian/.

Chapter 5: Fun with Hash Functions

Motivation

Purpose of Network Size Estimation

- Human curiosity
- Detection of unusual events
- Value of a botnet
- Tuning parameter

Functional Goals

- All peers obtain the network size estimate
- Supports churn
- Fully decentralized
- Efficient, secure with good load-balancing
- Operates in unstructured topologies
- Works well with modest clock skew between peers
- Ability to trade-off precision vs. efficiency

Intuitive Idea

- Set of elements distributed in a space
- Pick a random spot
- Measure distance to nearest element
- More elements $\Rightarrow$ smaller distance, more *overlapping*

Intuitive Idea

Intuitive Idea

Intuitive Idea

Intuitive Idea

Intuitive Idea - Applied to networks

- Space: all possible IDs
- Population: randomly distributed peer IDs
- Overlap: number of leading bits in common with a random ID

Theorem

Let $\overline{p}$ be the expected maximum number of leading overlapping bits between all n random node identifiers in the network and a random key. Then the network size n is approximately

$$2^{\overline{p}}$$

- $1 \Rightarrow 2$
- $6 \Rightarrow 64$
- $22 \Rightarrow 4$ M

Theorem

Let $\overline{p}$ be the expected maximum number of leading overlapping bits between all n random node identifiers in the network and a random key. Then the network size n is

$$2^{\overline{p}-0.332747}$$

- $1 \Rightarrow$ 1-2
- $6 \Rightarrow 50$
- $22 \Rightarrow$ 3.3 M

Our Approach: Key Points

- Use the current time to generate a random number
- More overlapping bits $\Rightarrow$ gossip earlier
- Also delay gossip randomly to avoid traffic spikes
- Proof-of-Work to make Sybil attacks harder
- Implemented! ($\approx$ 1500 lines C code in GNUnet)

Security

Attacker Model

- Freely participate
- Multiple identities
- May alter, drop, send/receive data
- Same resources as "normal" peers

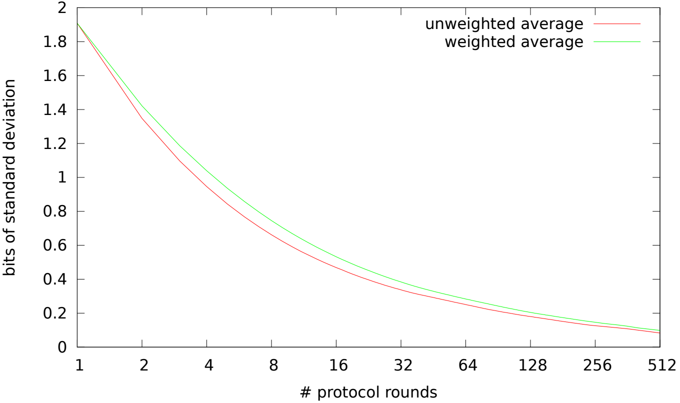
Security Properties

- Resistant to malicious participants (DoS, Manipulation)
- No trusted third parties
- Reliable

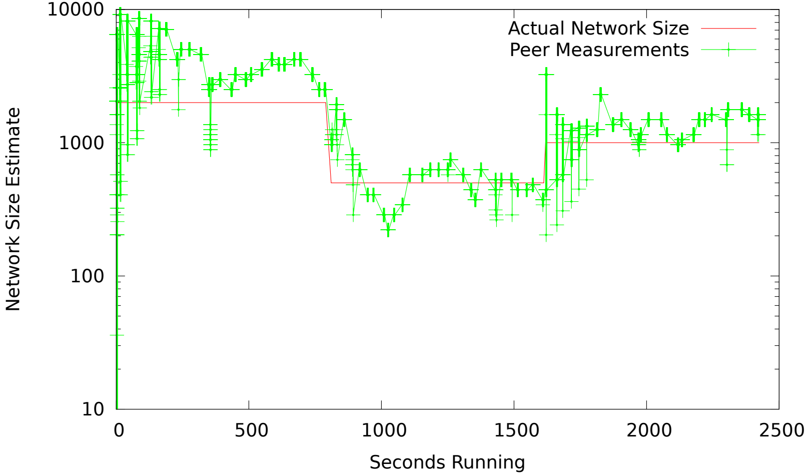
Processing results

- Final agreed value fluctuates around the actual size
- Last *i* protocol rounds are analyzed
  - Weighted average
  - Standard Deviation
- Precision - Cost tradeoff

Precision vs. Rounds of Measurement



Agreement between peers



Conclusion

- Mathematical foundation applicable broadly for group size estimates
- Secure & Efficient Network Size Estimation Protocol
- Arbitrary Topologies, Clock Skew harmless, DoS resistant