Distributed Key Generation and Threshold Decryption for OpenPGP

Heiko Stamer

HeikoStamer@gmx.net

76F7 3011 329D 27DB 8D7C 3F97 4F58 4EB8 FB2B E14F

HeikoStamer.dkg@gmx.net

B02D 1F23 0D4F 78F8 09B5 ABA0 2D18 CACE 1FA4 F2B4

26. Krypto-Tag, June 2017, SUSE, Nürnberg

Background

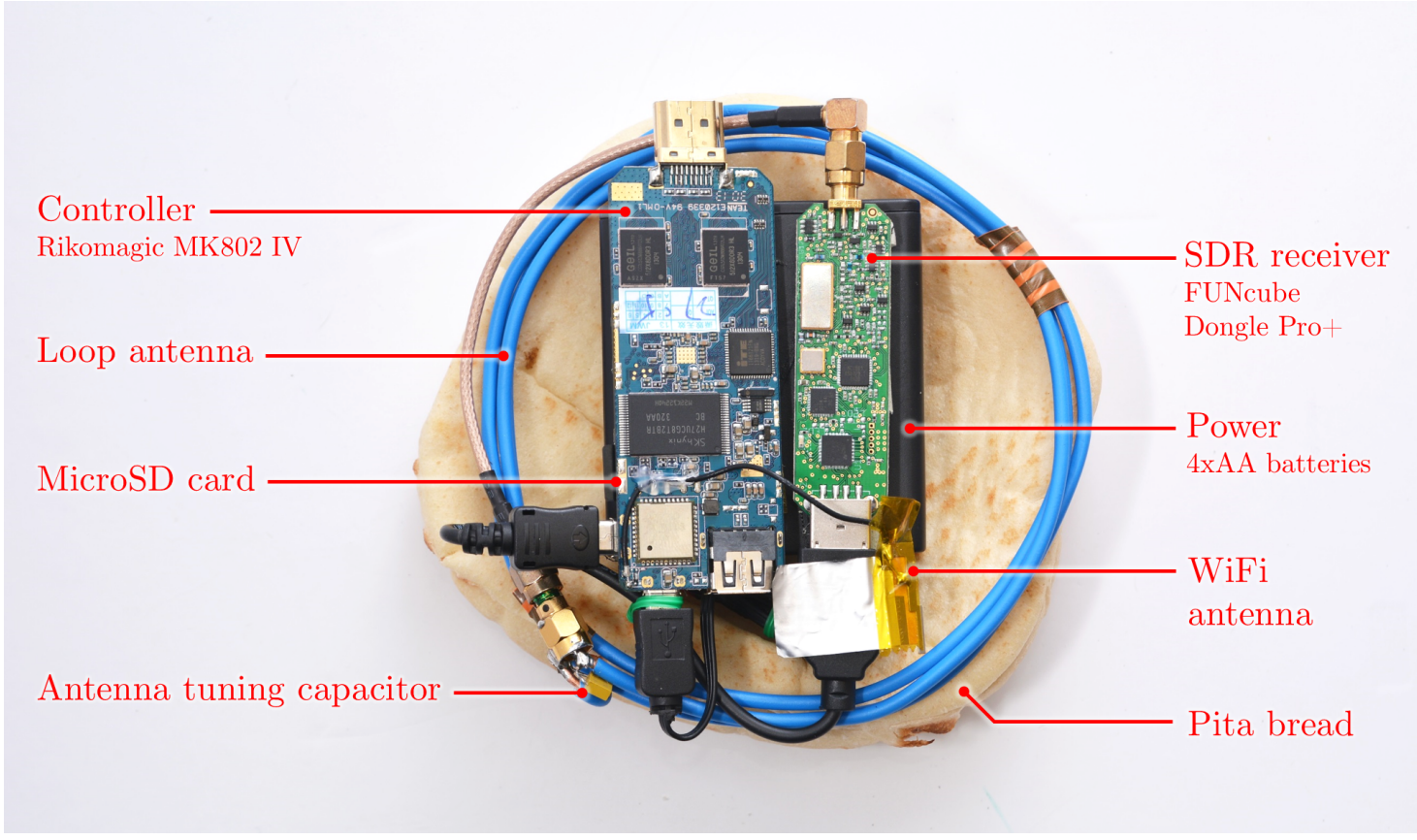
Source: Bruno Sanchez-Andrade Nuño, CC BY 2.0

Phillip Rogaway: *The Moral Character of Cryptographic Work*
`http://web.cs.ucdavis.edu/~rogaway/papers/moral.html`

> *We need to realize popular services in a secure, distributed, and decentralized way, powered by free software and free/open hardware.*

What is the problem?

Where is the problem?



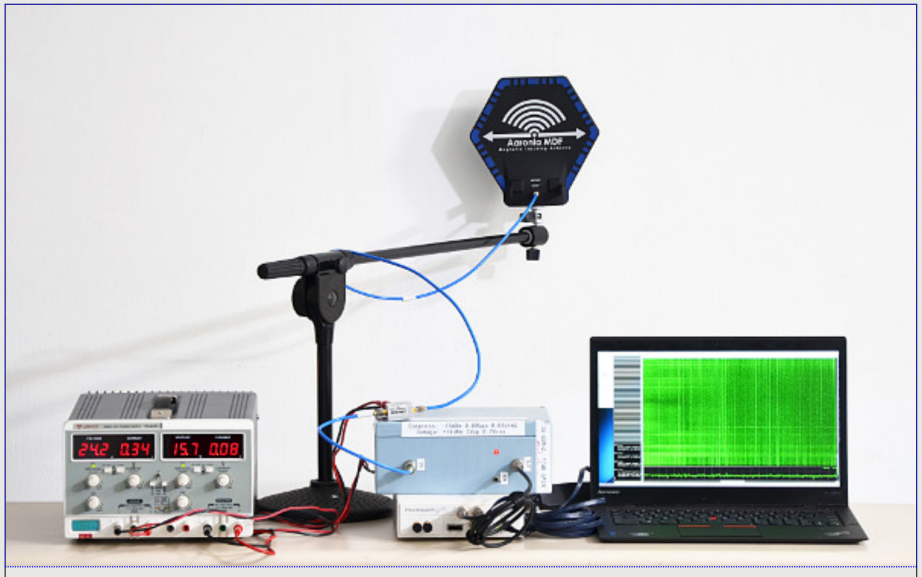
Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer.
Stealing Keys from PCs using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation. http://eprint.iacr.org/2015/170

Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2015.

Vulnerable software: GnuPG $\leqslant$ 1.4.18, Libgcrypt $\leqslant$ 1.6.2 (CVE-2014-3591)

Where is the problem?

Better side-channel attacks on ECDH and ECDSA followed . . .



Attacker's setup for capturing EM emanations. Left to right: power supply, antenna on a stand, amplifiers, software defined radio (white box), analysis computer.

Target laptop performing ECDH decryption using its secret key , on the other side of the wall.

Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer.
ECDH key-extraction via low-bandwidth electromagnetic attacks on PCs.
`https://eprint.iacr.org/2016/129`
RSA Conference Cryptographers' Track (CT-RSA) 2016.
Costs: \$ 3000, Vulnerable software: Libgcrypt $\leqslant$ 1.6.3 (CVE-2015-7511)

Mitigation measures

Make side-channel attacks difficult

- Hardware: electromagnetic shielding or tamper-proof HSM
- Software: constant-time operations on secret key material

Splitting/Sharing of private keys

- Example ICANN/IANA: DNSSEC root zone signing key

  `https://www.cloudflare.com/dns/dnssec/root-signing-ceremony/`

  `https://www.iana.org/dnssec/ceremonies/`
- Example Debian GNU/Linux: FTP archive signing key

  `https://ftp-master.debian.org/keys.html`

  `http://www.digital-scurf.org/software/libgfshare`

  The program gfshare (package libgfshare-bin) (a Shamir's secret sharing scheme implementation) is used to produce 5 shares of which 3 are needed to recover the secret key.

Problems: trusted hardware needed, more side-channels issues possible (e.g. CVE-2016-6316), no verifiable secret sharing (VSS)

Distributed Key Generation (DKG)

GJKR07 Gennaro, Jarecki, Krawczyk, Rabin: *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems.* JoC 20(1) 2007.

Preliminaries: set of n parties $P_1, \ldots, P_n$ with *partially synchronous* communication (e.g. synchronized clocks)

Assumptions:

- computing discrete logarithms modulo large primes is hard
- let p and q big primes such that $q \mid p-1$; then G_q denotes the subgroup of elements from $\mathbb{Z}_p^*$ of order q and g, h are generators of G_q such that $\log_g h$ is not known to adversary

Adversary:

- can corrupt up to t parties, where $t < n/2$ (optimal threshold or t-*resilience* for a synchronous model)
- is *static*, i.e., chooses corrupted parties at the beginning
- is *rushing*, i.e., speaks last in each round of communication

Distributed Key Generation (DKG)

GJKR07 Gennaro, Jarecki, Krawczyk, Rabin: *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems.* JoC 20(1) 2007.

Security: A DKG protocol is called t-*secure*, if in presence of an attacker that corrupts at most t parties the following requirements for correctness and secrecy are satisfied:

(C1) all subsets of $t+1$ shares provided by honest parties define the same unique secret key $x \in G_q$,

(C2) all honest parties have the same public key $y = g^x \bmod p$, where x is the unique secret guaranteed by (C1),

(C3) x is uniformly distributed in G_q,

(S1) no information on x can be learned by the adversary except for what is implied by $y = g^x \bmod p$ ($\exists \mathcal{S}$: PPT simulator).

Distributed Key Generation (DKG)

GJKR07 Gennaro, Jarecki, Krawczyk, Rabin: *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems.* JoC 20(1) 2007.

Security: A DKG protocol is called t-*secure*, if in presence of an attacker that corrupts at most t parties the following requirements for correctness and secrecy are satisfied:

(C1) all subsets of $t+1$ shares provided by honest parties define the same unique secret key $x \in G_q$,

(C2) all honest parties have the same public key $y = g^x \bmod p$, where x is the unique secret guaranteed by (C1),

(C3) x is uniformly distributed in G_q,

(S1) no information on x can be learned by the adversary except for what is implied by $y = g^x \bmod p$ ($\exists \mathcal{S}$: PPT simulator).

Robustness: construction of y and reconstruction of x is possible in presence of $\leqslant t$ malicious parties that try to foil computation

Efficient Distributed Key Generation

GJKR07 Gennaro, Jarecki, Krawczyk, Rabin: *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems.* JoC 20(1) 2007.

Security: A DKG protocol is called t-*secure*, if in presence of an attacker that corrupts at most t parties the following requirements for correctness and secrecy are satisfied:

(C1') there is an efficient procedure that on input the n shares submitted by the parties and the public information produced by the DKG protocol, outputs the unique value $x \in G_q$, even if up to t shares are submitted by faulty parties,

(C2) all honest parties have the same public key $y = g^x \bmod p$, where x is the unique secret guaranteed by (C1'),

(C3) x is uniformly distributed in G_q,

(S1) no information on x can be learned by the adversary except for what is implied by $y = g^x \bmod p$ ($\exists \mathcal{S}$: PPT simulator).

Protocol New-DKG [GJKR07]

Generating $x = \sum_{i \in QUAL} z_i \bmod q$:

1. Each party P_i performs Pedersen-VSS of secret z_i as a dealer
   - **(a)** Choose random polynomials $f_i(z) = a_{i0} + a_{i1} z + \cdots + a_{it} z^t$ and $f'_i(z) = b_{i0} + b_{i1} z + \cdots + b_{it} z^t$ over $\mathbb{Z}_q$, let $z_i = a_{i0} = f_i(0)$, broadcast commitment $C_{ik} = g^{a_{ik}} h^{b_{ik}} \bmod p$ for $k = 0, \ldots, t$, and send shares $s_{ij} = f_i(j) \bmod q$ and $s'_{ij} = f'_i(j) \bmod q$ to party P_j
   - **(b)** Each party P_j verifies that $g^{s_{ij}} h^{s'_{ij}} = \prod_{k=0}^{t} (C_{ik})^{j^k} \bmod p$
   - **(c), (d)** Resolution of received complaints from verification of the shares
2. Each party builds the set QUAL (non-disqualified parties)
3. Each party P_i computes secret share as $x_i = \sum_{j \in QUAL} s_{ji} \bmod q$

Extracting $y = g^x \bmod p$: (only non-disqualified parties, i.e., $i \in QUAL$)

4. Each party P_i exposes $y_i = g^{z_i} \bmod p$ via Feldman-VSS:
   - **(a)** Each party P_i broadcasts $A_{ik} = g^{a_{ik}} \bmod p$ for $k = 0, \ldots, t$
   - **(b)** Each party P_j verifies that $g^{s_{ij}} = \prod_{k=0}^{t} (A_{ik})^{j^k} \bmod p$
   - **(c)** Run reconstruction to compute $z_i, f_i(z), A_{ik}$, if P_i corrupted

   Set $y_i = A_{i0} = g^{z_i} \bmod p$ and compute $y = \prod_{i \in QUAL} y_i \bmod p$

Threshold Decryption (ElGamal Cryptosystem)

CGS97 Cramer, Gennaro, Schoenmakers: *A Secure and Optimally Efficient Multi-Authority Election Scheme.* EUROCRYPT 1997.

Encryption: message $m \in G_q$ is encrypted as $(g^k, y^k m)$, where $y \in G_q$ is the public key and secret $k \in \mathbb{Z}_q$ is chosen randomly

Decryption:

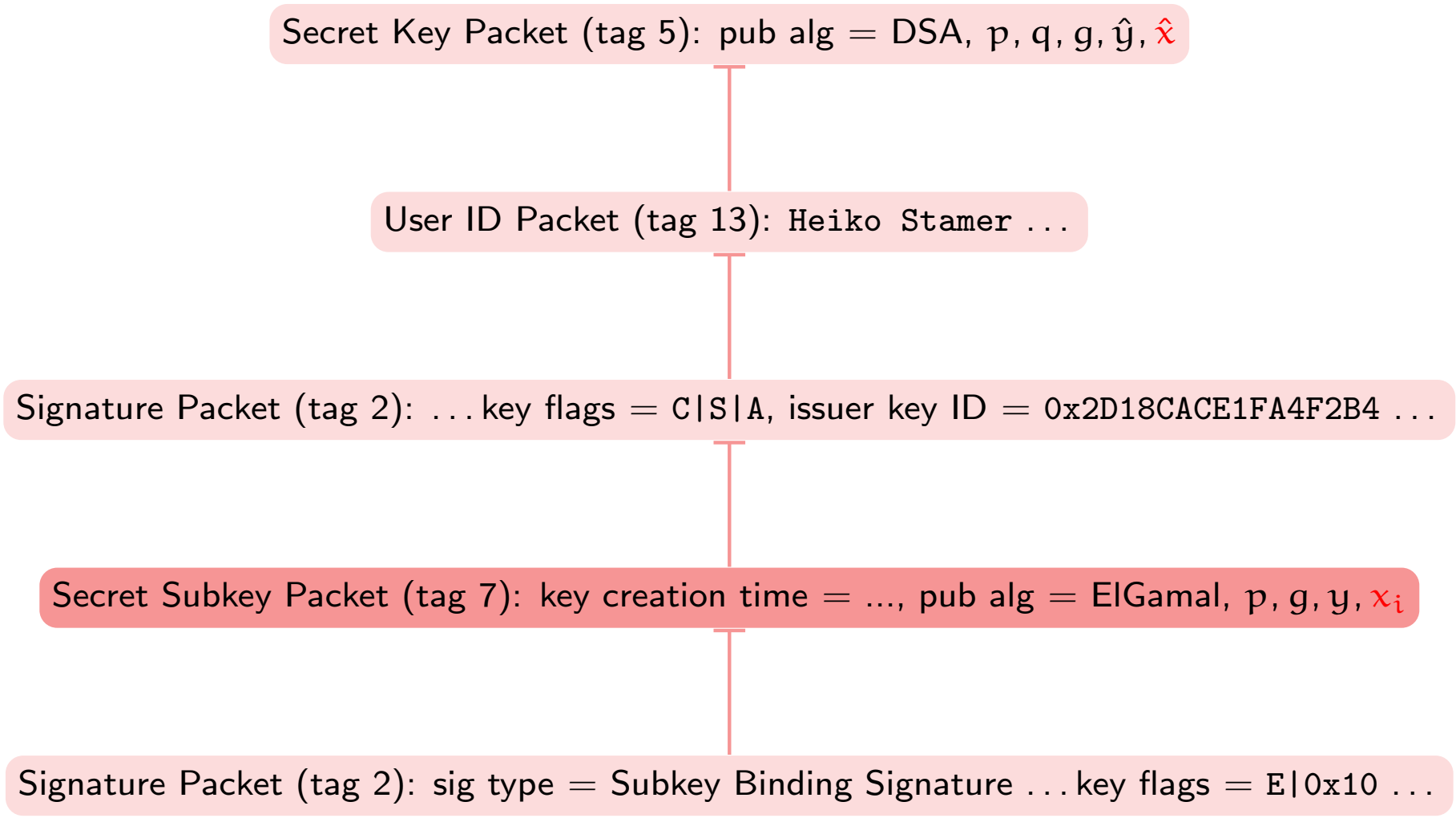
1. Each party P_i broadcast decryption share $r_i = (g^k)^{x_i} \bmod p$ together with a *zero-knowledge proof of knowledge* that $\log_g v_i = \log_{(g^k)} r_i$ holds, where $v_i = g^{x_i} \bmod p$ is a public verification key that can be computed after New-DKG 4.(a):

$$v_i = \prod_{j \in \mathsf{QUAL}} \prod_{k=0}^{t} (A_{jk})^{i^k} \bmod p$$

2. Combine $t+1$ correct decryption shares by using Lagrange interpolation in exponent: $m = (y^k m) / \prod_{j \in \Lambda} r_j^{\lambda_{j,\Lambda}} \bmod p$

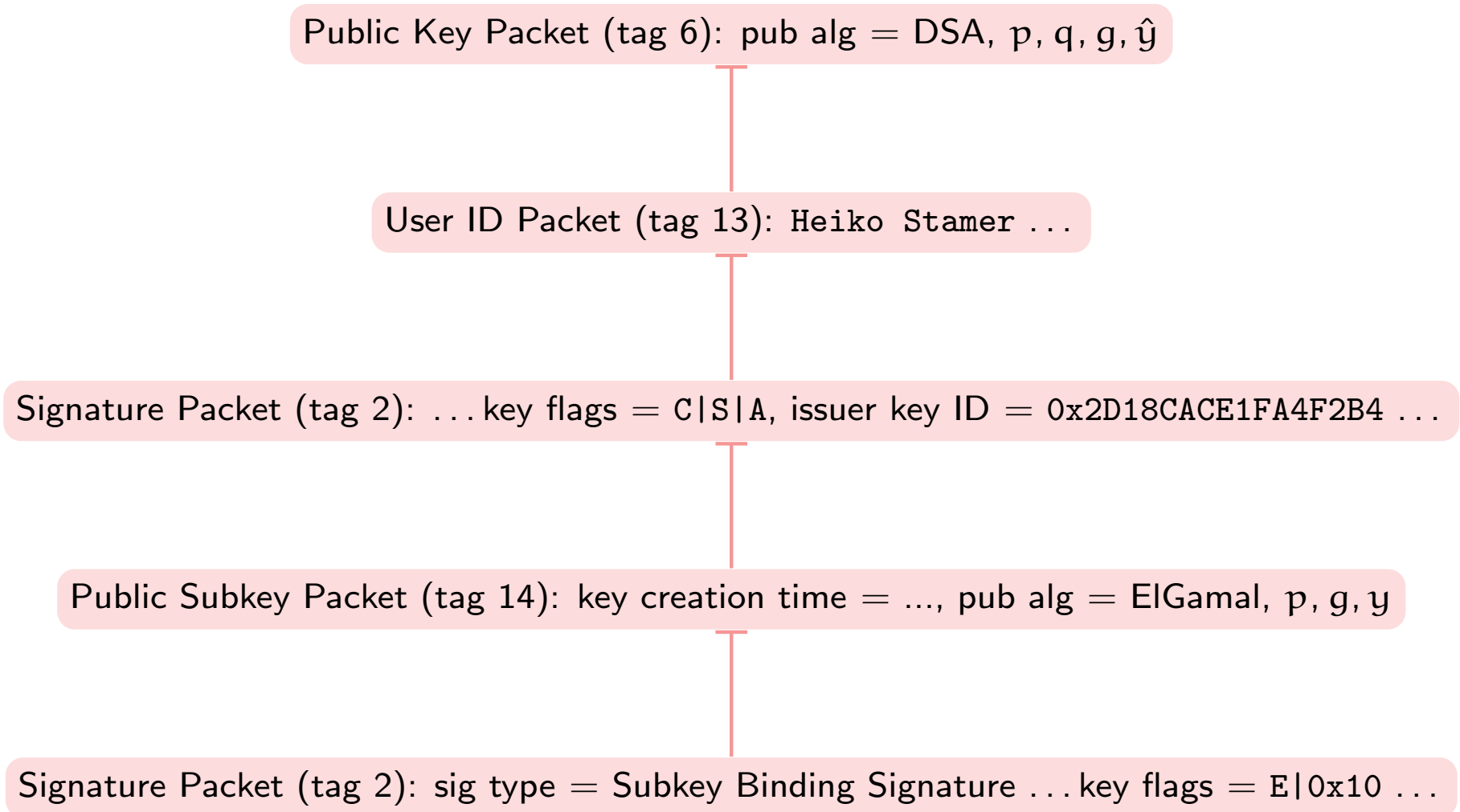
Implementation for OpenPGP [RFC4880]

Current State: Each party P_i has its individual primary DSA key (for signing etc.) and a common ElGamal subkey (for encryption)



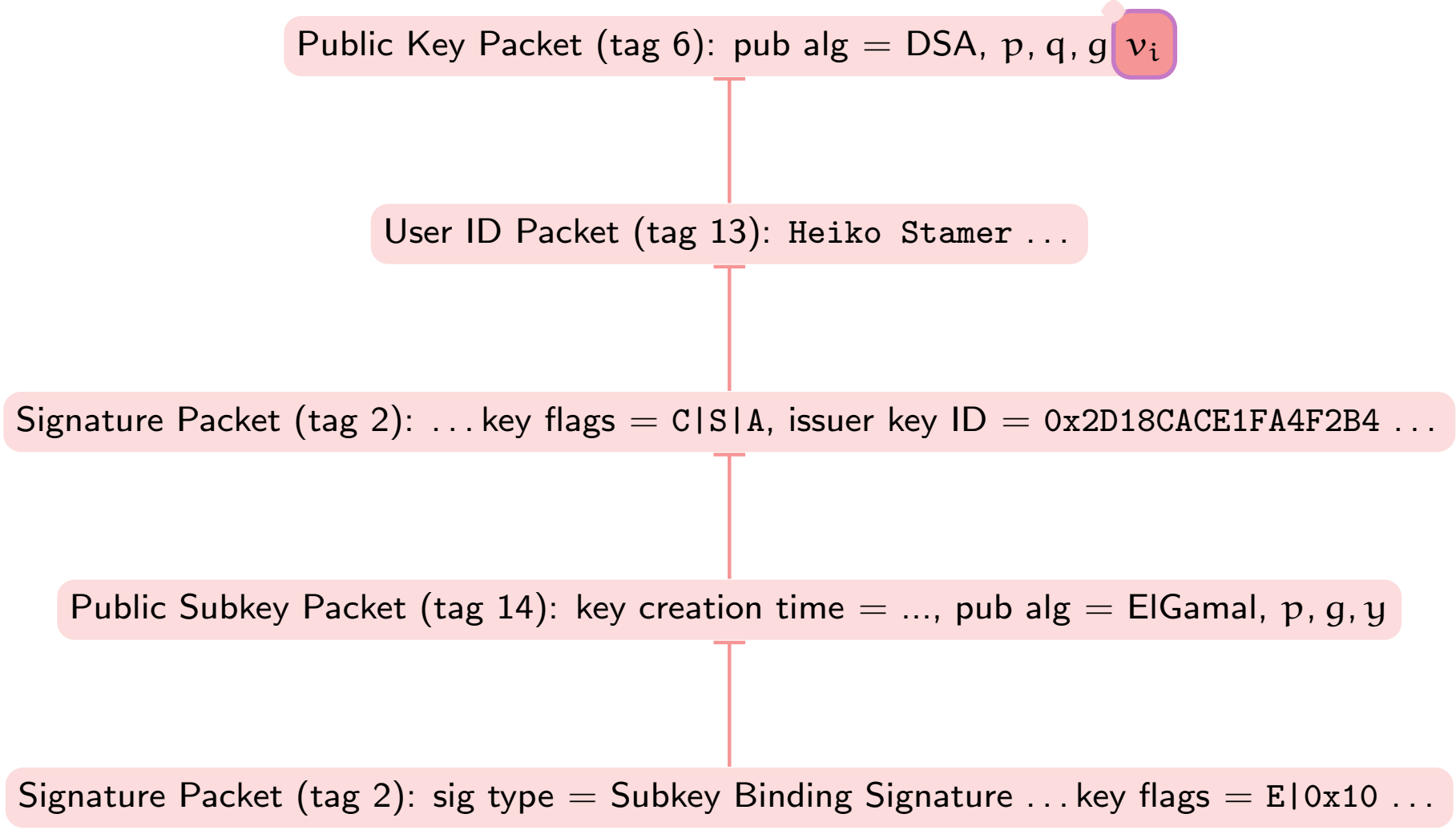
But where to store index i and verification key v_i?

Current State: Each party P_i has its individual primary DSA key (for signing etc.) and a common ElGamal subkey (for encryption)



Reusing x_i is probably a bad idea. Isn't it?

Current State: Each party P_i has its individual primary DSA key (for signing etc.) and a common ElGamal subkey (for encryption)



Implementation in LibTMCG

WARNING: Code is in EXPERIMENTAL state and should not be used for production!

New-DKG: `GennaroJareckiKrawczykRabinDKG.cc`
contains $\approx$ 1.600 LOC (incl. New-TSch)

Reliable Broadcast: `CachinKursawePetzoldShoupSEABP.cc`
contains $\approx$ 700 LOC; RBC Protocol [CKPS01] for $t < n/3$

OpenPGP: `CallasDonnerhackeFinneyShawThayerRFC4880.cc`
contains $\approx$ 2.800 LOC

3rd Party Libaries:

- GNU Multiple Precision Arithmetic Library (`libgmp`) $\geqslant$ 4.2.0
- GNU Crypto Library (`libgcrypt`) $\geqslant$ 1.6.0 (random, crypto primitives)

P2P Message Exchange: GNUnet $\geqslant$ 0.10.2 (not yet released!)

`dkg-gencrs` `dkg-generate` `dkg-encrypt` `dkg-decrypt`

Usage scenarios

Mailbox for informants/whistleblowers: *freedom of press*

- Imagine a newspaper or broadcast media with n responsible journalists in the editorial department
- There are authenticated private channels (e.g. already exchanged OpenPGP keys) between the journalists
- At least $t + 1$ of these journalists should be necessary to decrypt messages received in this dedicated mailbox

Shared mailbox for groups of political activists:

- Similar scenario as above

Protection of encryption keys against (gov.) malware:

- Imagine n devices with different security level (e.g. OS)
- At least $t + 1$ of these devices storing key shares should work together in order to decrypt messages

Remaining work

Cryptographic Protocols/Schemes:

- Threshold signature scheme for DSS/DSA
  - Gennaro, Jarecki, Krawczyk, Rabin: *Robust Threshold DSS Signatures*. EUROCRYPT 1996.
  - Gennaro, Goldfeder, Narayanan: *Threshold-optimal DSA/ECDSA Signatures and an Application to BitCoin Wallet Security*. ACNS 2016.
- Interactive variant of verifying decryption shares to stay in so-called *standard model* without random oracle assumption
- $\mathfrak{h}$-generation protocol with *distributed zero-knowledge PoKs*
- Adaptive security (add a zero-knowledge PoK in step 4. of New-DKG)

Software Engineering:

- Package (`dkg-openpgp`?) containing only the DKG tools
- Full compliance with OpenPGP standard
- Fully asynchronous communication model without artifical timing assumptions, cf. related work [KG09, KHG12]
- State-based representation of New-DKG
- Generic group abstraction layer in LibTMCG (e.g. for ECC)

How can you help?

- Compiling and testing LibTMCG on different platforms
- Review the design criterias and invent new usage scenarios
- Review the source code and report vulnerabilities/bugs
- Help with implementation of missing protocols (e.g. RSA, ECC)
- Packaging for different distributions of free/libre software
- Design and advocate for including ν_i's in revised RFC 4880bis

References

GJKR07 Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin.
Secure Distributed Key Generation for Discrete-Log Based Cryptosystems.
Journal of Cryptology, 20(1):51–83, 2007.

CGS97 Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers.
A Secure and Optimally Efficient Multi-Authority Election Scheme.
Advances in Cryptology — EUROCRYPT '97, LNCS 1233, pp. 103–118, 1997.

CKPS01 Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup.
Secure and Efficient Asynchronous Broadcast Protocols.
Advances in Cryptology — CRYPTO '01, LNCS 2139, pp. 524–541, 2001.

KG09 Aniket Kate and Ian Goldberg.
Distributed Key Generation for the Internet.
Proceedings of ICDCS 2009, pp. 119–128, 2009.

KHG12 Aniket Kate, Yizhou Huang, and Ian Goldberg.
Distributed Key Generation in the Wild.
Cryptology ePrint Archive: Report 2012/377, 2012.
`https://eprint.iacr.org/2012/377`

RFC4880 J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer.
OpenPGP Message Format.
Network Working Group, Request for Comments, No. 4880, November 2007.

Sta17 Heiko Stamer.
LibTMCG. Version $\geqslant$ 1.3.0. `http://www.nongnu.org/libtmcg/`