Distributed Key Generation and Threshold Cryptography for OpenPGP

Heiko Stamer

HeikoStamer@gmx.net
76F7 3011 329D 27DB 8D7C 3F97 4F58 4EB8 FB2B E14F

HeikoStamer.dkg@gmx.net
9EBD C46A B510 F909 21DB 84B2 DD28 EE5A E478 3280

Datengarten/81, October 2017, Berlin

Background

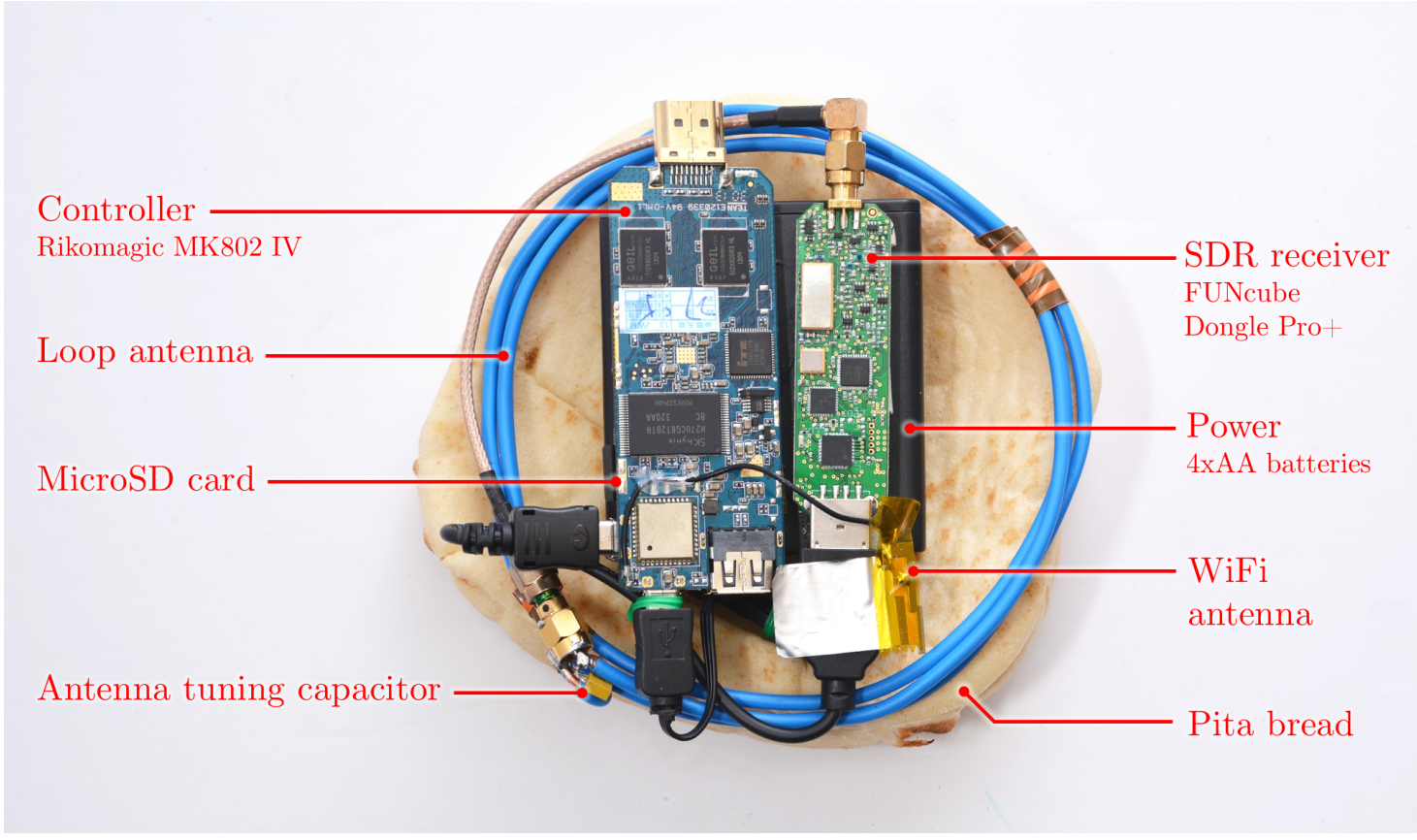
Source: Bruno Sanchez-Andrade Nuño, CC BY 2.0

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?



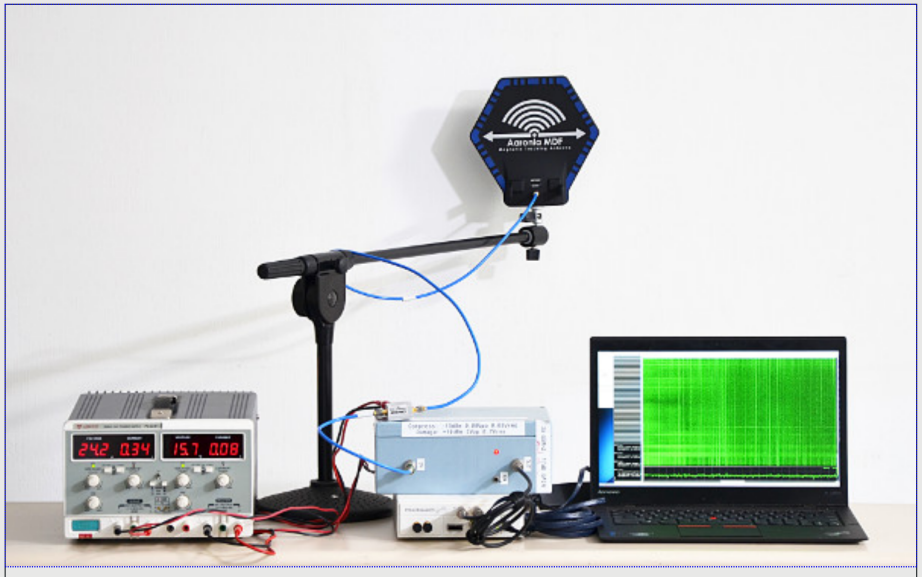
Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer.
Stealing Keys from PCs using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation. http://eprint.iacr.org/2015/170

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.

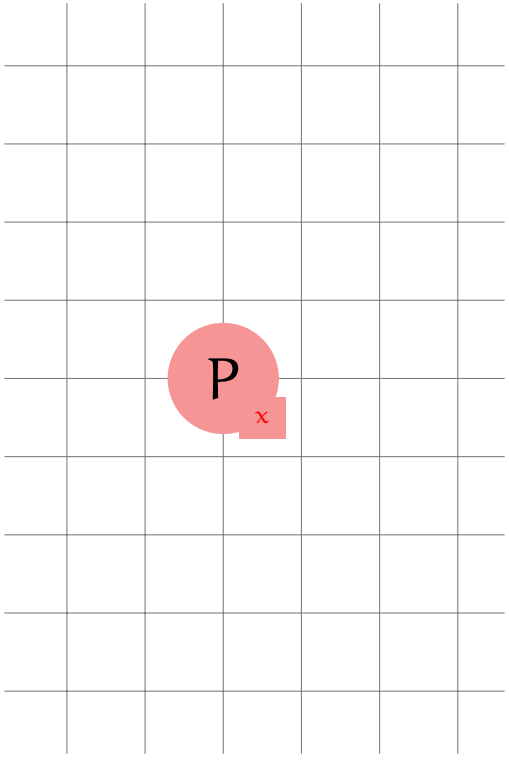
Problems: trusted hardware needed, more side-channels issues possible (e.g. CVE-2016-6316), no verifiable secret sharing (VSS)

Threshold Cryptography

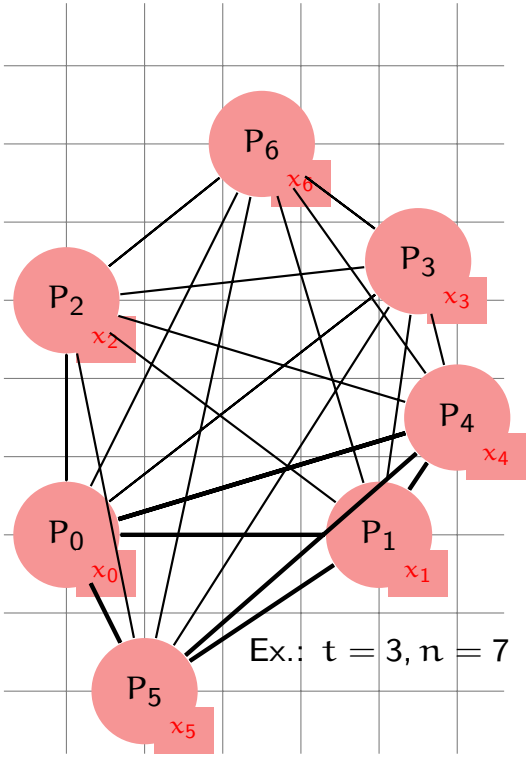
Boy86 Boyd: *Digital Multisignatures.* Cryptography and Coding, 1986.

Des87 Desmedt: *Society and Group Oriented Cryptography: A New Concept.* CRYPTO 1987.

DF89 Desmedt, Frankel: *Threshold Cryptosystems.* CRYPTO 1989.



one secret and single-party algorithms (Generate, Decrypt, Sign)



shared secret and distributed algorithms with threshold $t < n$

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

Adversary:

- is *malicious*; 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

Properties of 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 $\mathcal{A}$ 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 (i.e. not corrupted by $\mathcal{A}$) 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 key guaranteed by (C1),

(C3) x is uniformly distributed in G_q,

(S1) no information on x can be learned by the adversary $\mathcal{A}$, except for what is implied by the public key $y = g^x \bmod p$

Properties of 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 $\mathcal{A}$ 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 (i.e. not corrupted by $\mathcal{A}$) 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 key guaranteed by (C1),

(C3) x is uniformly distributed in G_q,

(S1) no information on x can be learned by the adversary $\mathcal{A}$, except for what is implied by the public key $y = g^x \bmod p$

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

Protocol New-DKG [GJKR07]

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

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_ℓ, $f_\ell(z)$, $A_{\ell k}$, if P_ℓ 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 corresponding public key and $k \overset{R}{\in} \mathbb{Z}_q$ a fresh secret

Decryption:

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

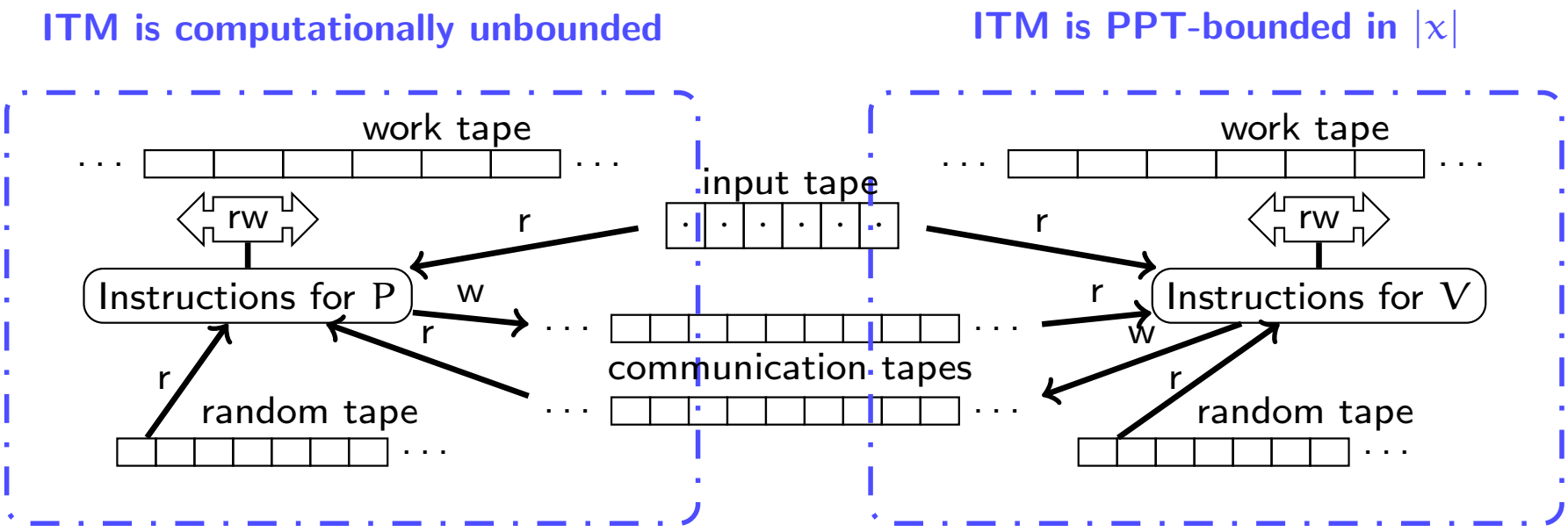
$$v_i = \prod_{j \in \mathsf{QUAL}} \prod_{k=0}^{t} (A_{jk})^{i^k} \bmod p$$

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

Interactive Proof Systems

GMR85 Goldwasser, Micali, Rackoff: *The Knowledge Complexity of Interactive Proof Systems.* STOC 1985. (SIAM J. Comput. 18(1) 1989)

Probabilistic *Interactive Proof System* (IP) for a statement $x \in L$

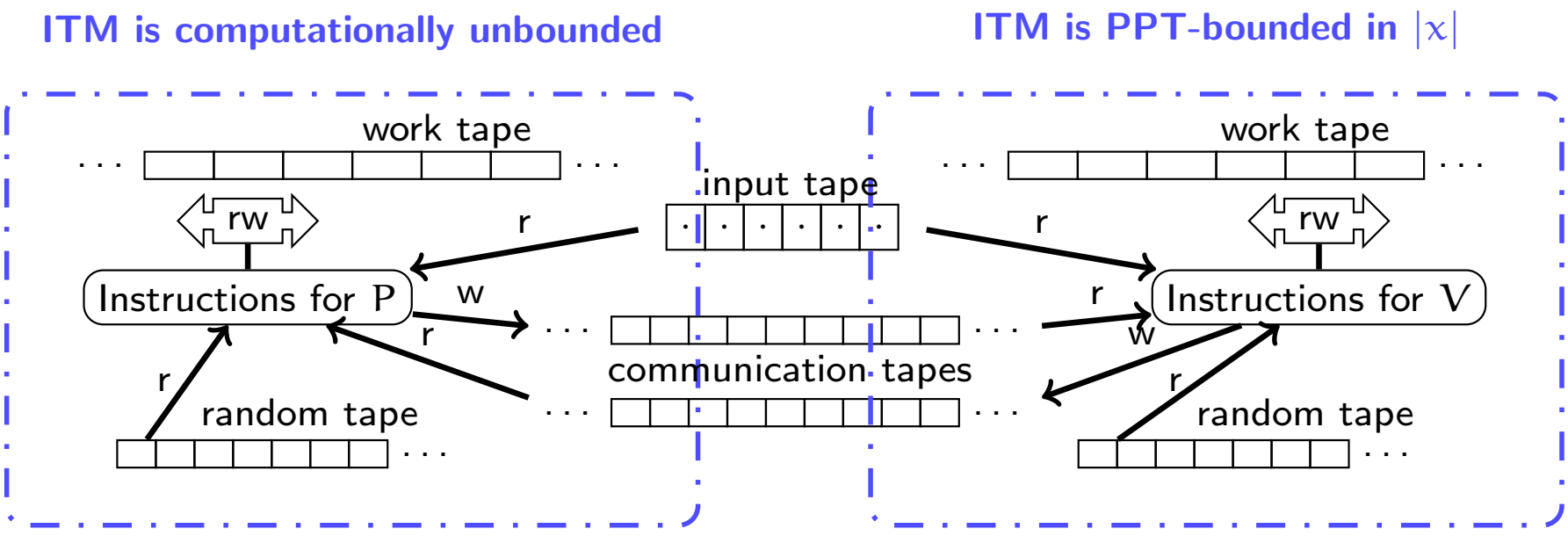


Completeness: if the statement is true, the honest verifier V will be convinced of this fact by an honest prover P

Soundness: if the statement is false, no cheating prover P can convince the honest verifier V that it is true, except with some small probability (soundness error)

Zero-Knowledge Proof

Probabilistic *Interactive Proof System* (IP) for a statement $x \in L$



Zero-Knowledge: if the statement is true, no cheating verifier V learns anything other than the fact that $x \in L$

Theorem (Goldreich, Micali, Wigderson 1986; Ben-Or et al. 1988)

$\mathcal{NP} \subseteq \mathcal{IP}_{\mathrm{CZK}}$, if one-way functions exist; $\mathcal{IP} = \mathcal{IP}_{\mathrm{CZK}}$, if one-way functions exist.

Theorem (Shamir 1990)

$\mathcal{IP} = \mathcal{PSPACE}$.

Example: ZK Proof of Graph Isomorphism $\in \mathcal{IP}_{\mathsf{PZK}}$

input: graphs G_1, G_2, **statement:** $G_1 \cong G_2$, **secret:** π s.t. $G_1 = \pi(G_2)$

P: $\sigma \overset{R}{\in} \{1,2\}$, $\psi \overset{R}{\in} \Pi(G_\sigma)$, compute $H = \psi(G_\sigma)$, send H to V

V: (challenge) $\tau \overset{R}{\in} \{1,2\}$, send τ to P

P: compute $\rho = \left\{ \begin{array}{ll} \psi & \text{if } \tau = \sigma \\ \psi \circ \pi & \text{if } \tau \neq \sigma \text{ and } \sigma = 1 \\ \psi \circ \pi^{-1} & \text{if } \tau \neq \sigma \text{ and } \sigma = 2 \end{array} \right]$, send ρ to V

V: check whether $H = \rho(G_\tau)$ holds and accept resp. reject

Completeness: honest prover P can always construct ρ s.t. $H = \rho(G_\tau)$

Soundness: error prob. $1/2$ (can be reduced by sequential repetitions)

Zero-Knowledge: $\forall \mathcal{V} : \exists \mathcal{S}$ (simulator, expected PPT) with identically distributed output as the view of the above protocol

(simulator $\mathcal{S}$ picks $\sigma' \overset{R}{\in} \{1,2\}, \psi' \overset{R}{\in} \Pi(G_{\sigma'})$, computes $H' = \psi'(G_{\sigma'})$ and outputs transcript (H', τ', ψ'), if $\mathcal{V}$'s challenge $\tau = \sigma'$, otherwise restart)

Zero-Knowledge Proof of Knowledge

GMR85 Goldwasser, Micali, Rackoff: *The Knowledge Complexity of Interactive Proof Systems.* STOC 1985.

FFS87 Feige, Fiat, Shamir: *Zero-Knowledge Proofs of Identity.* STOC 1987.

BG92 Bellare, Goldreich: *On Defining Proofs of Knowledge.* CRYPTO 1992.

$\mathrm{L} \in \mathcal{NP}$: show that P "knows" a corresponding short *witness* ω for proving membership of each $x \in \mathrm{L}$ without revealing these secrets

Definition (informal)

The protocol Π is a *Zero-Knowledge Proof of Knowledge* (ZKPoK), iff

1. Π is an Interactive Proof System with zero-knowledge property,
2. for any ITM $\mathcal{P}$ that make V accept the input x there exists a PPT-bounded *knowledge extractor* $\mathcal{M}$ that can rewind the execution of $\mathcal{P}$ (i.e. reset the head and content of work tape, the heads of input and random tape and the state of its finite control unit) and thus extract a witness ω showing membership $x \in \mathrm{L}$.

Σ-protocol: three-round ZKPoK (P: commitment, V: challenge, P: response)

Example: Equality of Discrete Logarithms (Σ-protocol)

CP92 Chaum, Pedersen: *Wallet Databases with Observers.* CRYPTO 1992.

Threshold Decryption [CGS97] (ElGamal Cryptosystem)

Let p and q be large primes such that $q \mid p-1$; then G_q denotes the unique subgroup of elements from $\mathbb{Z}_p^*$ of order q and g denotes a generator of G_q.

public verification key of P_i: $v_i = g^{x_i} \bmod p$

decryption share of P_i: $r_i = (g^k)^{x_i} \bmod p$

input: p, q, g, v_i, g^k, r_i, **statement:** $\log_g v_i = \log_{(g^k)} r_i \pmod p$

P: $s \overset{R}{\in} \mathbb{Z}_q$, commit to $(a, b) = (g^s, (g^k)^s)$, send (a, b) to V

V: (challenge) $c \overset{R}{\in} \mathbb{Z}_q$ and send c to P

P: compute $d = cx_i + s \bmod q$ and send d to V

V: accept, if $g^d = a(v_i)^c \pmod p$ and $(g^k)^d = b(r_i)^c \pmod p$

Knowledge Extractor: rewind $\mathcal{P}$ to get (c_1, d_1) and (c_2, d_2) for same s; since $c_1 \neq c_2$ it can compute $x_i = \frac{d_1 - d_2}{c_1 - c_2} = \frac{(c_1 x_i + s) - (c_2 x_i + s)}{c_1 - c_2} \bmod q$

Security of ElGamal in $\mathbb{Z}_p^*$ (e.g. in OpenPGP)

Sakurai, Shizuya: *Relationships among the Computational Powers of Breaking Discrete Log Cryptosystems.* EUROCRYPT 1995.

Sakurai, Shizuya: *A Structural Comparison of the Computational Difficulty of Breaking Discrete Log Cryptosystems.* JoC 11(1), 1998.

- Computing $m \in \mathbb{Z}_p^*$ from given $g, y, g^k, y^k m \in \mathbb{Z}_p^*$ is hard, iff the Computational Diffie-Hellman (CDH) problem is hard

$\rightsquigarrow$ ElGamal in $\mathbb{Z}_p^*$ is OW-CPA secure under CDH assumption

Tsiounis, Yung: *On the Security of ElGamal based Encryption.* PKC 1998.

- Distinguishing $m, \bar{m} \in G_q$ given $g, y, g^k, y^k m, g^{\bar{k}}, y^{\bar{k}} \bar{m} \in G_q$ is hard, iff the Decision Diffie-Hellman (DDH) problem is hard

$\rightsquigarrow$ ElGamal in G_q is IND-CPA secure under DDH assumption

Threshold Signature Scheme (DSA/DSS Variant)

CGJKR99 Canetti, Gennaro, Jarecki, Krawczyk, Rabin: *Adaptive Security for Threshold Cryptosystems.* CRYPTO 1999.

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

Adversary:

- can corrupt up to $\hat{t}$ parties, where $\hat{t} < n/2$ (optimal threshold or $\hat{t}$-*resilience* for a synchronous model)
- is *adaptive*, i.e., can choose corrupted parties during attack
- is *rushing*, i.e., speaks last in each round of communication

Protocol DL-Key-Gen (optimally-resilient) [CGJKR99]

Generating common secret $\widehat{x} = \sum_{i \in \widehat{QUAL}} \widehat{z}_i \bmod q$:

1. Parties execute Joint-RVSS (i.e. each P_i performs a Pedersen-VSS of random secret $\widehat{z}_i$ as a dealer) and get $\widehat{C}_{ik}$, $\widehat{QUAL}$, shares $\widehat{x}_i$, $\widehat{x}'_i$

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

2. Each party P_i broadcasts $\widehat{A}_i = g^{\widehat{z}_i} \bmod p$ and $\widehat{B}_i = h^{\widehat{f}'_i(0)} \bmod p$ such that $\widehat{C}_{i0} = \widehat{A}_i \cdot \widehat{B}_i \pmod p$ holds

3.–6. Each party P_i proves with a *distributed zero-knowledge proof of knowledge* that the above split of the commitment $\widehat{C}_{i0}$ is correct

7. Run reconstruction to compute $\widehat{z}_j$ and $\widehat{A}_j$, if some P_j are corrupted

8. The public value $\widehat{y}$ is set to $\widehat{y} = \prod_{i \in \widehat{QUAL}} \widehat{A}_i \bmod p$

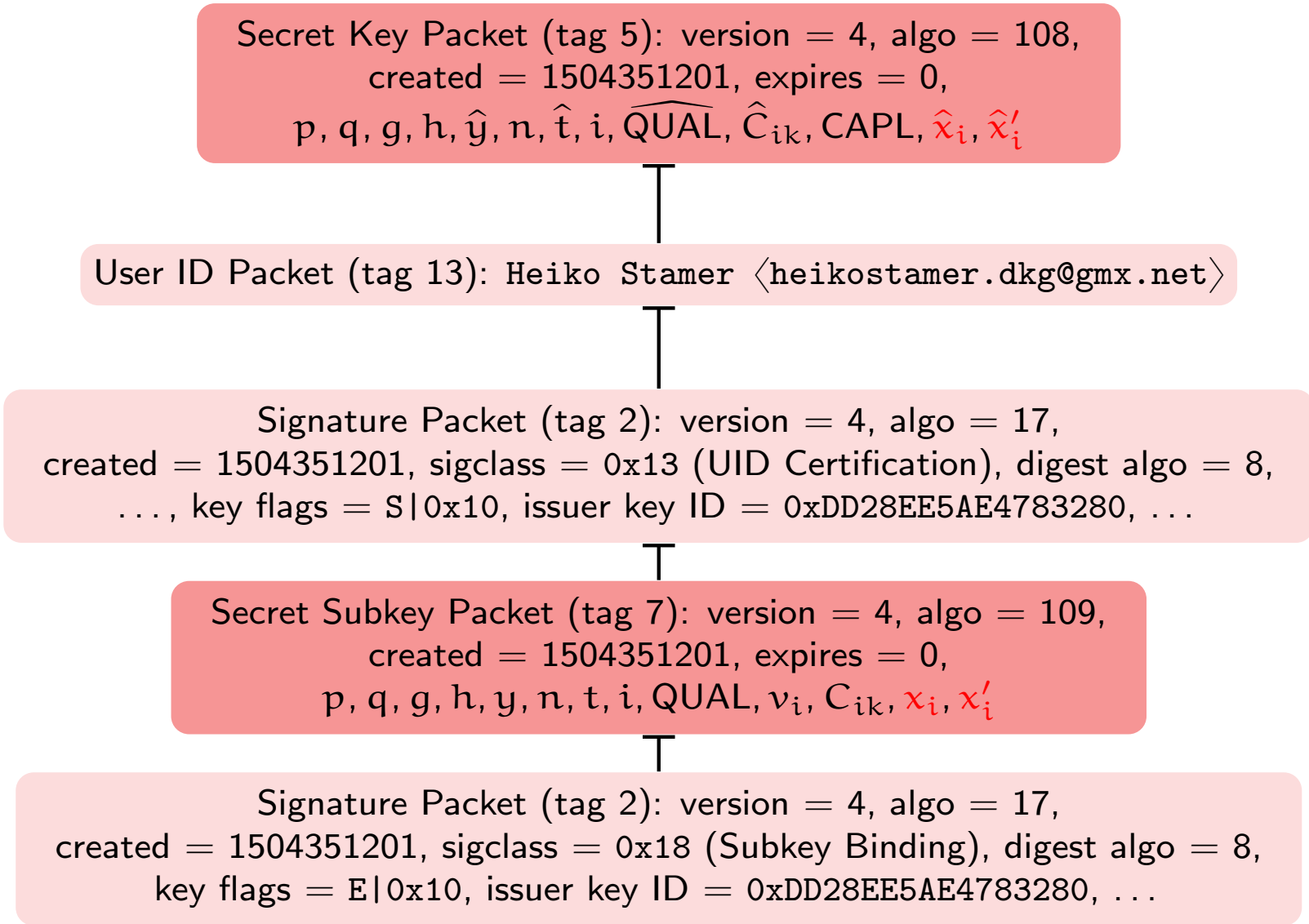
9. P_i erases all secrets generated in this protocol aside from $\widehat{x}_i$ and $\widehat{x}'_i$

Protocol DSS-Sig-Gen ($\geqslant 2\hat{t}+1$, not optimal) [CGJKR99]

1. Generate $r = g^{k^{-1}} \bmod p \bmod q$:
   - **(a)** Parties execute Joint-RVSS to generate k and get shares k_i, k'_i
   - **(b)** Parties execute DL-Key-Gen to generate a and get g^a and a_i, a'_i
   - **(c)** Back-up k_i and a_i using Pedersen-VSS; P_i is required to prove correctness with a *distributed zero-knowledge proof of knowledge* (at least $\hat{t}+1$ sound proofs and corrupted parties will be ignored)
   - **(d)** Each P_i shares $\hat{v}_i = a_i k_i \bmod q$ using Pedersen-VSS and proves correctness with a *distributed zero-knowledge proof of knowledge*
   - **(e)** Run reconstruction of a_j and k_j, if some P_j are corrupted, and set $\hat{v}_j = a_j k_j$; bad values are sieved out using commitments from (c)
   - **(f)** Each P_i broadcasts its shares of the $\hat{t}$-degree polynomial, which is a linear combination of the shares $\hat{v}_1, \ldots, \hat{v}_{2\hat{t}+1}$ received in step (d)
   - **(g)** Each P_i computes locally μ^{-1} and $r = (g^a)^{\mu^{-1}} \bmod p \bmod q$
2. Generate $s = k(m + \hat{x}r) \bmod q$:
   - Parties perform steps equivalent to 1.(c)–(f), with the values $m + \hat{x}_i r$ taking the role of a_i's, and with s taking the role of μ (in step 1.(c) only the back-up of $m + \hat{x}_i r$ is required; reuse k_i's)
3. Party P_i erases all secrets generated in this protocol

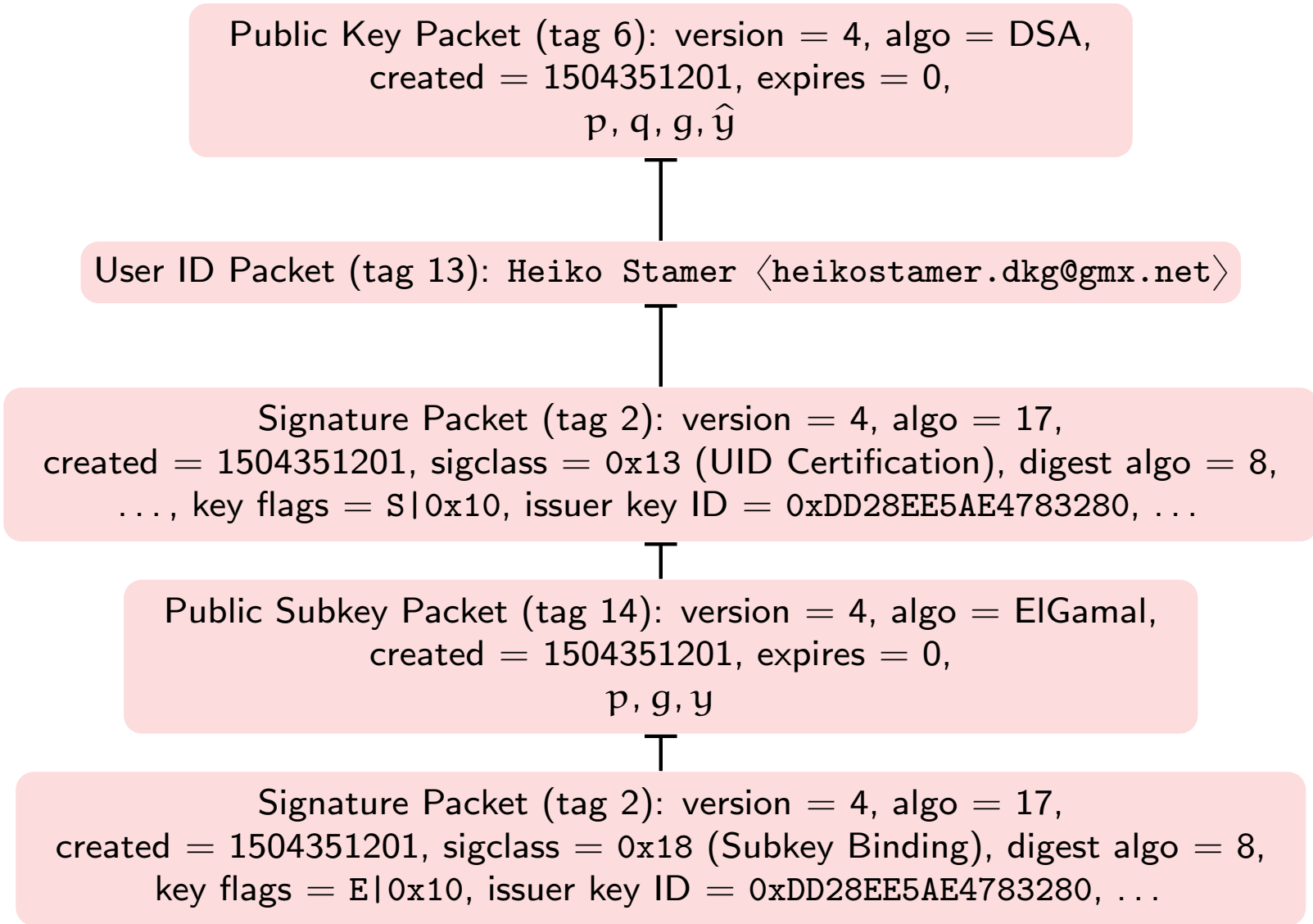
Implementation for OpenPGP [RFC4880]

Case 1: Each party P_i has a shared primary DSA key (for signing) and a shared ElGamal subkey (for encryption)



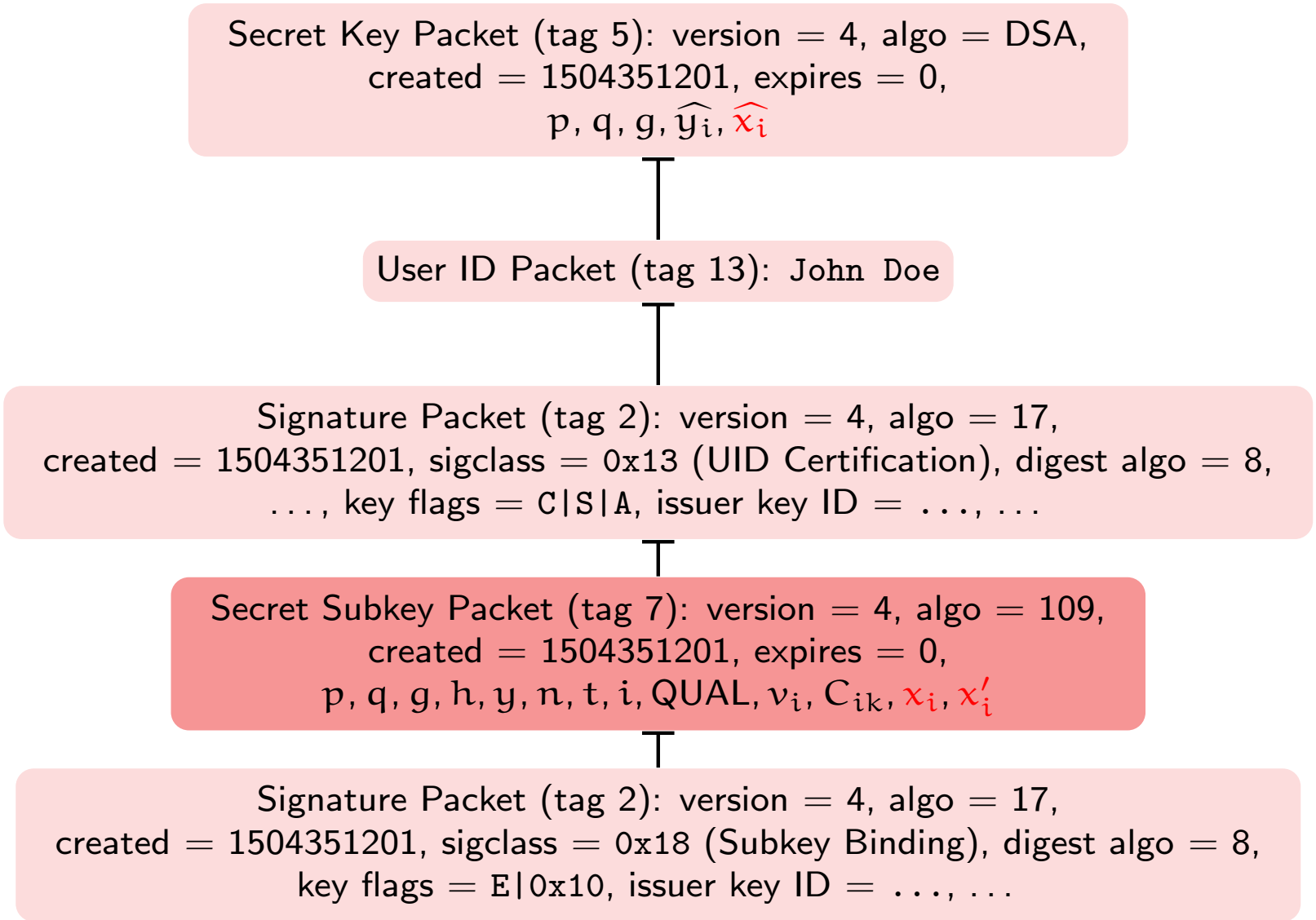
Corresponding OpenPGP-compatible Public Key

Case 1: Each party P_i has shared primary DSA key (for signing) and a shared ElGamal subkey (for encryption)



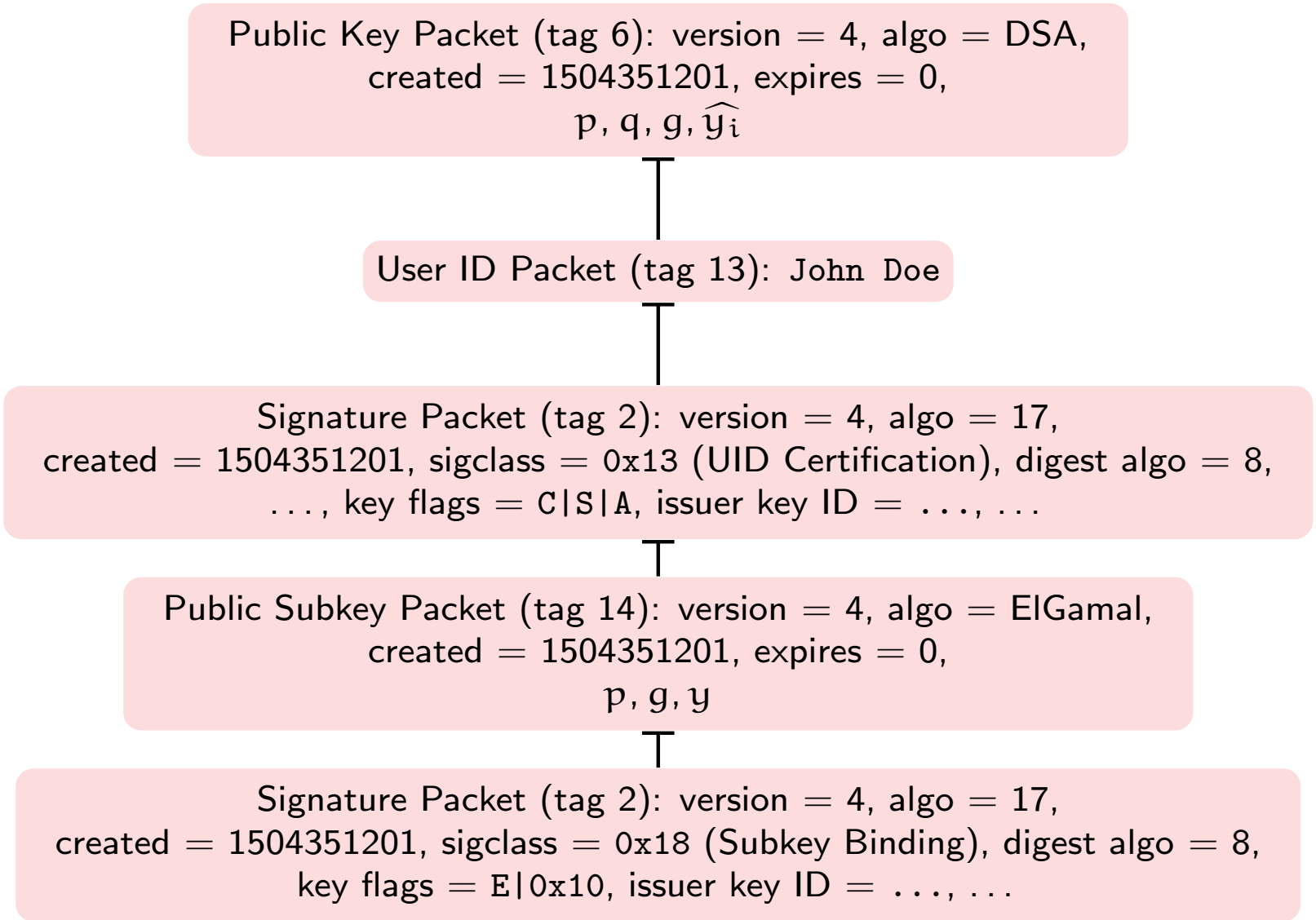
Other Cases

Case 2: Each party P_i has an individual primary DSA key (for signing etc.) and a shared ElGamal subkey (for encryption)



Corresponding OpenPGP-compatible Public Key

Case 2: Each party P_i has an individual primary DSA key (for signing etc.) and a shared ElGamal subkey (for encryption)



Other Cases

Case 3: Each party P_i has only a shared primary DSA key

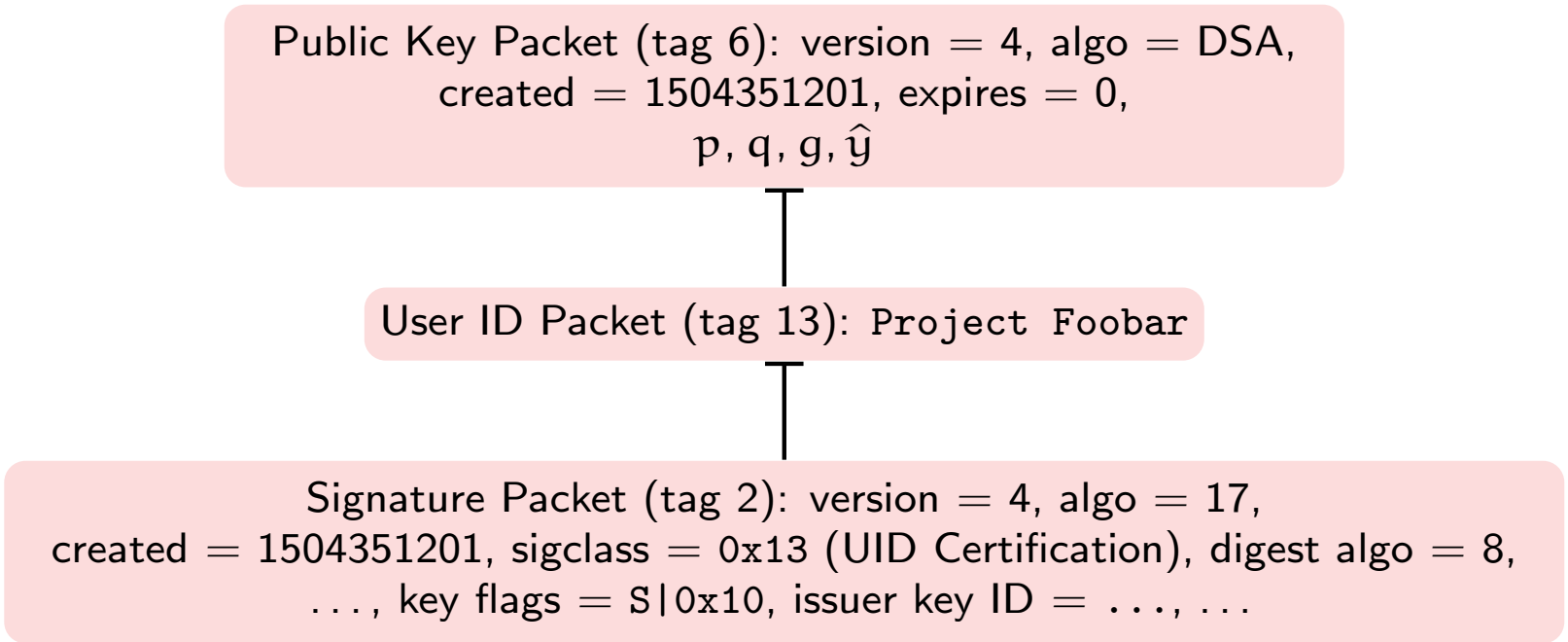
Secret Key Packet (tag 5): version $=$ 4, algo $=$ 108,
created $=$ 1504351201, expires $=$ 0,
$p, q, g, h, \widehat{y}, n, \widehat{t}, i, \widehat{QUAL}, \widehat{C}_{ik}$, CAPL, $\widehat{x}_i, \widehat{x}'_i$

User ID Packet (tag 13): `Project Foobar`

Signature Packet (tag 2): version $=$ 4, algo $=$ 17,
created $=$ 1504351201, sigclass $=$ `0x13` (UID Certification), digest algo $=$ 8,
$\ldots$, key flags $=$ `S|0x10`, issuer key ID $=$ $\ldots$, $\ldots$

Corresponding OpenPGP-compatible Public Key

Case 3: Each party P_i has only a shared primary DSA key



Implementation in LibTMCG resp. DKGPG

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

New-DKG, New-TSch:

`GennaroJareckiKrawczykRabinDKG.cc`

contains $\approx$ 1.750 LOC

Joint-RVSS, Joint-ZVSS, DL-Key-Gen, DSS-Sig-Gen:

`CanettiGennaroJareckiKrawczykRabinASTC.cc`

contains $\approx$ 4.500 LOC (+900 LOC `PedersenVSS.cc`)

Reliable Broadcast: `CachinKursawePetzoldShoupSEABP.cc`

contains $\approx$ 850 LOC; RBC Protocol [CKPS01] for $t < n/3$

OpenPGP: `CallasDonnerhackeFinneyShawThayerRFC4880.cc`

contains $\approx$ 3.650 LOC

3rd Party Libraries:

- 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!), TCP/IP interface (e.g. TOR hidden service with port forwarding and `torsocks`)

User Interface (DKGPG = Distributed Privacy Guard)

dkg-gencrs domain parameter generation (p, q, g) of G_q

- **-f SEED** generate domain parameters according to FIPS 186-4

dkg-generate distributed key generation (DSA+ElGamal)

- **-e TIME** expiration time of generated keys in seconds
- **-g STRING** domain parameters of G_q (common reference string)
- **-H STRING** hostname of this peer for TCP/IP (e.g. onion address)
- **-P STRING** password list to encrypt/authenticate TCP/IP connections
- **-s INTEGER** threshold $\hat{t}$ for DL-Key-Gen protocol (signature scheme)
- **-t INTEGER** threshold t for New-DKG protocol (encryption scheme)

dkg-decrypt threshold decryption (ElGamal)

- **-i FILENAME** input file with ASCII-armored encrypted message
- **-n** switch to non-interactive mode (using NIZK proofs; ROM)
- **-o FILENAME** output file with decrypted message

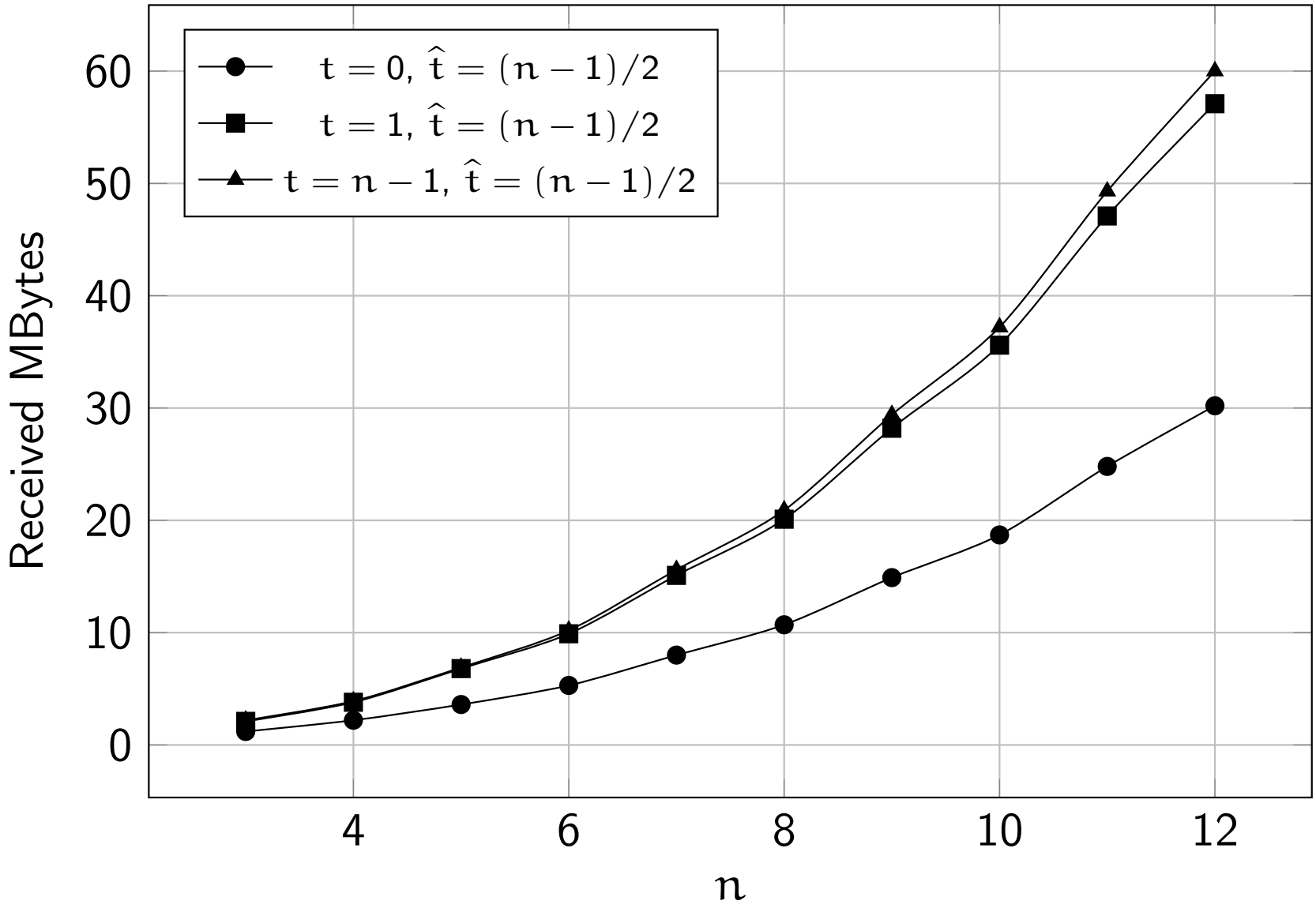
dkg-sign threshold signature generation (DSA)

- **-e TIME** expiration time of generated signature in seconds
- **-i FILENAME** create detached signature from given input file
- **-o FILENAME** output file with detached signature

dkg-revoke threshold key revocation (DSA+ElGamal)

- **-r INTEGER** reason for revocation (OpenPGP machine-readable code)

Network Traffic (`dkg-generate` with $|p| = 2048$, $|q| = 256$)



Usage Scenarios

Mailbox for informants/whistleblowers: *distributed power*

- Imagine a newspaper or broadcast media with n responsible journalists in the editorial department/board
- There are authenticated private channels (e.g. already exchanged GNUnet/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 with additional signing capability

Protection of encryption/signing keys of a single person:

- Imagine n devices with different security levels (e.g. OS)
- At least $t + 1$ resp. $2\widehat{t} + 1$ of these devices (storing the key shares) must work together to decrypt resp. sign messages

Remaining Work (TODO)

Cryptographic Protocols/Schemes:

- $\mathfrak{h}$-generation protocol with *distributed zero-knowledge PoKs*
- Proactive refresh of shares protects against mobile adversary

Software Engineering:

- Package (`dkgpg`) containing only the DKG tools
- Fully asynchronous communication model without artifical timing assumptions, cf. related work [KG09, KHG12]
- State-based representation of the protocols
- Generic group abstraction layer in LibTMCG (e.g. for ECC)

How can you help?

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

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.

CGJKR99 Ran Canetti, R. Gennaro, S. Jarecki, Hugo Krawczyk, and Tal Rabin.
Adaptive Security for Threshold Cryptosystems. (extended paper available)
Advances in Cryptology — CRYPTO '99, LNCS 1666, pp. 98–116, 1999.

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.