Verteilte Schlüsselerzeugung für OpenPGP

Distributed Privacy Guard (DKGPG)

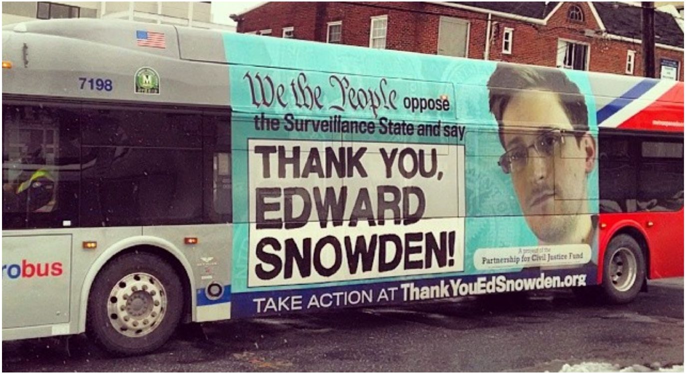
Heiko Stamer

HeikoStamer@gmx.net

9EBD C46A B510 F909 21DB 84B2 DD28 EE5A E478 3280

35C3, December 2018, Leipzig

Introduction



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.*

How to keep your private keys secret?

1. Encrypt private key material (e.g. RFC4880: S2K mechanism)
2. Make side-channel attacks difficult
   - Hardware: electromagnetic shielding or tamper-proof HSM
   - Software: constant-time operations on private key material
3. 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`
     `https://git.gitano.org.uk/libgfshare.git/`

     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.

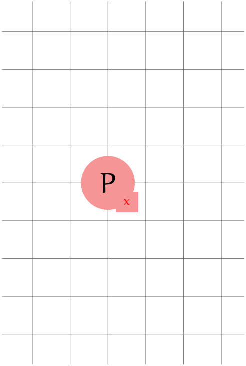
Problems: weak S2K, trusted hardware needed, side-channel issues still possible, no verifiable secret sharing (VSS), combine step

Threshold Cryptography

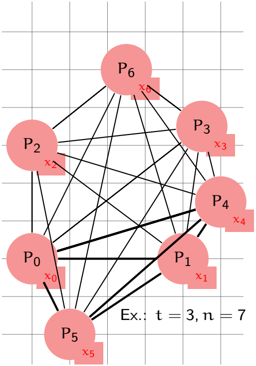
Boyd: *Digital Multisignatures.* Cryptography and Coding, 1986.

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

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

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

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

Threshold Cryptography for OpenPGP [RFC4880]

Basic Case: Each P_i has a shared primary DSA key (for signing) and one [or more] shared ElGamal subkey[s] (for decryption)

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

User ID Packet (tag 13): `Heiko Stamer` ⟨`heikostamer@gmx.net`⟩

Signature Packet (tag 2): version $=$ 4, algo $=$ 17,
created $=$ 1541534836, sigclass $=$ `0x13` (UID Certification), digest algo $=$ 8, ...
key flags $=$ `C|S|0x10`, issuer key ID $=$ `0xDD28EE5AE4783280`, ..., issuer fpr v4

Secret Subkey Packet (tag 7): version $=$ 4, algo $=$ 109,
created $=$ 1504351201, expires $=$ 0,
p, q, g, h, y, n, t, i, QUAL, v_i, C_{ik}, x_i, x'_i

Signature Packet (tag 2): version $=$ 4, algo $=$ 17,
created $=$ 1504351201, sigclass $=$ `0x18` (Subkey Binding), digest algo $=$ 8, ...
key flags $=$ `E|0x10`, issuer key ID $=$ `0xDD28EE5AE4783280`, ...

Corresponding OpenPGP-compatible Public Key

Basic Case: All parties have a common primary DSA key (for verification) and common ElGamal subkey[s] (for encryption)

Public Key Packet (tag 6): version = 4, algo = DSA,
created = 1504351201, expires = 0,
$p, q, g, \widehat{y}$

User ID Packet (tag 13): `Heiko Stamer` ⟨`heikostamer@gmx.net`⟩

Signature Packet (tag 2): version = 4, algo = 17,
created = 1541534836, sigclass = `0x13` (UID Certification), digest algo = 8, ...
key flags = `C|S|0x10`, issuer key ID = `0xDD28EE5AE4783280`, ..., issuer fpr v4

Public Subkey Packet (tag 14): version = 4, algo = ElGamal,
created = 1504351201, expires = 0,
p, g, y

Signature Packet (tag 2): version = 4, algo = 17,
created = 1504351201, sigclass = `0x18` (Subkey Binding), digest algo = 8, ...
key flags = `E|0x10`, issuer key ID = `0xDD28EE5AE4783280`, ...

Threshold Cryptography for OpenPGP [RFC4880]

Sign-Only Case: Each party P_i has a shared primary DSA key

Secret Key Packet (tag 5): version = 4, algo = 108,
created = 1504345345, expires = 31536000,
$p, q, g, h, \hat{y}, n, \hat{t}, i, \widehat{\text{QUAL}}, \hat{C}_{ik}, \text{CAPL}, \hat{x}_i, \hat{x}'_i$

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

Signature Packet (tag 2): version = 4, algo = 17,
created = 1504345345, sigclass = `0x13` (UID Certification), digest algo = 8, ...
key flags = `C|S|0x10`, issuer key ID = ..., ..., issuer fpr v4

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. $2\hat{t} + 1$ of these devices (storing the key shares) must work together to decrypt resp. sign messages

LibTMCG: C++ Classes for Schemes/Protocols

WARNING: Code is still EXPERIMENTAL and SHOULD NOT be used for production!

New-DKG, New-TSch:

`GennaroJareckiKrawczykRabinDKG.cc`

contains ≈ 1.800 LOC

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

`CanettiGennaroJareckiKrawczykRabinASTC.cc`

contains ≈ 4.900 LOC (+900 LOC `PedersenVSS.cc`)

OpenPGP: `CallasDonnerhackeFinneyShawThayerRFC4880.cc`

contains ≈ 16.100 LOC

3rd Party Libraries/Dependencies:

- GNU Multiple Precision Arithmetic Library (`libgmp`) ⩾ 4.2.0
- GNU Crypto Library (`libgcrypt`) ⩾ 1.6.0 (random, crypto primitives)
- GNU Privacy Guard Error Code Library (`libgpg-error`) ⩾ 1.12
- ⋆ Botan: Crypto and TLS for C++11 (`libbotan-2`) ⩾ 2.x (random)

DKGPG: Bunch of Command-Line Programs

WARNING: It's still EXPERIMENTAL and SHOULD NOT be used for production!

Status: β-version 1.1.0 released at 08-Dec-2018, ≈ 21.800 LOC

Dependencies:

- Toolbox for Mental Card Games (`libTMCG`) ⩾ 1.3.16
- GNU Multiple Precision Arithmetic Library (`libgmp`) ⩾ 4.2.0
- GNU Crypto Library (`libgcrypt`) ⩾ 1.6.0
- GNU Privacy Guard Error Code Library (`libgpg-error`) ⩾ 1.12
- zlib Compression Library (`libz`) ⩾ 1.2.3
- ⋆ Library for Data Compression (`libbzip2`) ⩾ 1.0.6

P2P Message Exchange:

- ⋆ CADET service of GNUnet ⩾ 0.11 (not yet released!)
- TCP/IP (e.g. TOR hidden service with port forwarding + `torsocks`)

Runs: Gentoo Linux, Debian GNU/Linux, FreeBSD, OpenBSD

Packages: OpenSuSE, Arch Linux (AUR)

User Interface: Distributed Key Generation

dkg-gencrs generate the domain parameters (p, q, g) of $\mathbb{G}_q$

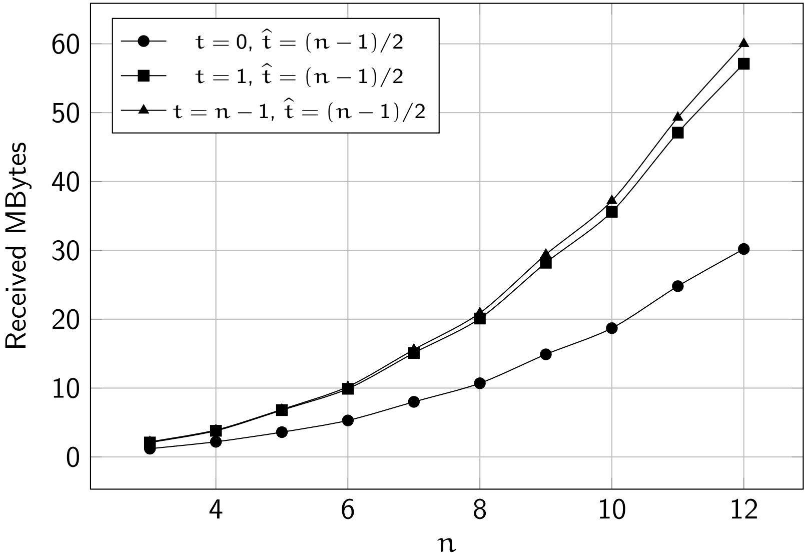
- **-f SEED_{62}** choose parameters according to FIPS 186-4 with SEED

dkg-generate distributed key generation (DSA±ElGamal)

- **-e INTEGER** expiration time of generated key[s] in seconds (default: 0)
- **-g STRING** domain parameters of $\mathbb{G}_q$ ("common reference string")
  default: fixed $\mathbb{G}_q$ with $|p| = 3072$ bit and $|q| = 256$ bit
  (Note that mathematical properties of $\mathbb{G}_q$ reveal DKGPG usage!)
- **-H STRING** hostname of the calling 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)
  default: $(n - 1)/2$
  range: $0, \ldots, (n - 1)/2$, non-shared primary keys by -s 0
- **-t INTEGER** threshold t for New-DKG protocol (encryption scheme)
  default: $(n - 1)/2$
  range: $0, \ldots, (n - 1)$, no encryption subkey by -t 0
- **-w INTEGER** minutes to wait until start of key generation (only GNUnet)
- **-W INTEGER** timeout for point-to-point messages in minutes (default: 5)
- **-y** yet another OpenPGP tool (generate a non-shared key pair)

dkg-addrevoker add external revocation key (cf. RFC 4880)

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



User Interface: Encryption and Decryption

dkg-encrypt message encryption with fixed cipher AES-256

-a INTEGER	enforce use of AEAD algorithm (cf. draft RFC 4880bis)
-b	write output in binary format instead of ASCII-armored
-i FILENAME	read message from a file instead of STDIN
-k FILENAME	keyring containing the required public keys
-o FILENAME	write encrypted output rather to file than STDOUT
-r	select key[s] from given keyring by KEYSPEC
-s STRING	select only encryption-capable subkeys with this fingerprint
-t	throw included key IDs for somewhat improved privacy
-w	allow weak keys

dkg-decrypt message decryption with two operational modes

-b	read input in binary format instead of ASCII-armored
-H STRING	hostname of this peer for TCP/IP (e.g. onion address)
-i FILENAME	read message from a file instead of STDIN
-k FILENAME	verify included signatures based on key[s] from keyring
-K	allow weak keys to verify included signatures
-n	switch to non-interactive mode (using NIZK proofs; ROM)
-o FILENAME	write decrypted output rather to file than STDOUT
-P STRING	password list to encrypt/authenticate TCP/IP connections
-w INTEGER	minutes to wait until start of decryption (only GNUnet)
-W INTEGER	timeout for point-to-point messages in minutes (default: 5)
-y FILENAME	yet another OpenPGP tool (use a non-tElG private key)

User Interface: Generate and Verify Signatures

dkg-verify verification of a single detached signature

-b	read input (i.e. KEYFILE and signature) in binary format
-f TIMESPEC	signature made before given time specification is not valid
-i FILENAME	read signed document from given file (mandatory option)
-k FILENAME	verify signature based on key from keyring instead of KEYFILE determined by issuer (fingerprint) subpacket from signature
-s FILENAME	read detached signature from file instead of STDIN
-t TIMESPEC	signature made after given time specification is not valid
-w	allow weak or expired keys

dkg-sign generation of a (detached) document signature

-C	apply cleartext signature framework (cf. RFC 4880)
-e INTEGER	expiration time of generated signature in seconds (default: 0)
-H STRING	hostname of this peer for TCP/IP (e.g. onion address)
-i FILENAME	read document to sign from given file (mandatory option)
-o FILENAME	write signature rather to file than STDOUT
-P STRING	password list to encrypt/authenticate TCP/IP connections
-t	create a canonical text document signature (cf. RFC 4880)
-U STRING	policy URI tied to generated signature
-w INTEGER	minutes to wait until start of decryption (only GNUnet)
-W INTEGER	timeout for point-to-point messages in minutes (default: 5)
-y FILENAME	yet another OpenPGP tool (use a non-tDSS private key)

User Interface: Miscellaneous Functions (1)

dkg-keysign certification signature generation

- **-1** issuer has not done any verification of the claim of identity
- **-2** issuer has done some casual verification of the claim of identity
- **-3** issuer has done substantial verification of the claim of identity
- **-e INTEGER** expiration time of generated signature in seconds (default: 0)
- **-r** create a certification revocation signature
- **-u STRING** sign only valid user IDs containing this string
- **-U STRING** policy URI tied to generated signature
- **-y FILENAME** yet another OpenPGP tool (use a non-tDSS private key)

dkg-adduid adds another user ID

- **-u STRING** the user ID to add (mandatory option)
- **-y FILENAME** yet another OpenPGP tool (use a non-tDSS private key)

dkg-revuid revokes a specified user ID

- **-u STRING** specifies the user ID to revoke (mandatory option)
- **-y FILENAME** yet another OpenPGP tool (use a non-tDSS private key)

dkg-revoke revocation (certificate) for a key (DSA$\pm$ElGamal)

- **-r INTEGER** reason for revocation (OpenPGP machine-readable code)
- **-R STRING** reason for revocation (human-readable form)

User Interface: Miscellaneous Functions (2)

dkg-keyinfo shows public data of a private key share

-m OLD NEW migrate peer identity (must keep lexicographical order of CAPL)

dkg-keycheck checks a public key for vulnerabilities (e.g. ROCA)

-r check only valid subkeys

dkg-refresh provides 'proactive security' (refresh of key shares)

dkg-timestamp generates a timestamp signature

-a include an OpenPGP notation that represents time deviation
-i FILENAME read the target signature from a file (mandatory option)
-s KEY:VALUE include an OpenPGP notation (e.g. serial number)
-y FILENAME yet another OpenPGP tool (use a non-tDSS private key)

dkg-timestamp-verify verification of a timestamp signature

-b read input (i.e. KEYFILE and signature) in binary format
-f TIMESPEC signature made before given time specification is not valid
-k FILENAME verify signature based on key from keyring instead of KEYFILE
-o FILENAME write the embedded target signature to a file instead of STDOUT
-s FILENAME read timestamp signature from file instead of STDIN
-t TIMESPEC signature made after given time specification is not valid
-w allow weak or expired keys

How can you help?

- Compiling and testing the software on different platforms
- Packaging for more distributions of free operating systems
- Review source code and report vulnerabilities/bugs
- Review design criterias and invent new usage scenarios
  Geer, Yung: *Split-and-Delegate: Threshold Cryptography for the Masses*. International Conference on Financial Cryptography 2002.
- Help with implementation of missing protocols (e.g. RSA, ECC)
- Switch to asynchronous communication model [KG09, KHG12]
- Write standardization draft and advocate for including threshold cryptography in revised RFC 4880bis or other
  NIST Project Threshold Cryptography: draft published, workshop March 2019

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.