Non-Abelian Groups as Candidate Platform for Cryptographic Schemes: Strengths and Weaknesses

Rainer Steinwandt

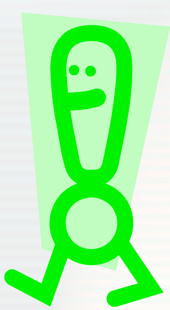
Florida Atlantic University

Non-abelian groups in cryptography: Why?

... besides being academic fun:

- **desire for new hardness assumptions**
  (not amenable to known quantum algorithms)

- **desire for performance improvements**
  (e.g., smaller signatures, faster encryption)



Other promising candidates exist, incl.

- Lattices ($\rightarrow$ NTRU)
- Multivariate Cryptography ($\rightarrow$ SFLASH)

... current state – a pragmatic view

- **many proposals** for encryption, signing, key establishment, ... using non-abelian groups **have been made**
- **most suggestions** that were specified in detail **have been attacked successfully**

hardly any non-abelian proposal available that is accepted as **practical and secure**

What is a cryptographic scheme?

Common problems in “non-abelian proposals”:

- lack of clearly specified assumptions & goals
  ➡ **security model unclear**
- **available** cryptographic **tools & attack models not taken into account**
- **lack of formal rigor**

“attacks w/o mathematical value”

Example: public key encryption

Established minimum requirement:

indistinguishable encryptions under **chosen plaintext attacks (with proof!)**

... or be more efficient than the others ☺

- encrypt one bit at a time
- “sufficiently complicated” instances
- hard to find plaintext from ciphertext

MST_1 revisited

- introduced as public key encryption scheme
- **no security proof**
- no secure key generation known

... could yield trapdoor one-way perm.
from a group-theoretical problem

combination with known cryptographic constructions could yield a “real scheme”

Logarithmic signatures of finite groups

G a finite group, $A_1, \ldots, A_s \subseteq G$ with $G = A_1 \cdot \ldots \cdot A_s$.

Then $[A_1, \ldots, A_s]$ is a **logarithmic signature** for G :⇔ each $g \in G$ has a **unique** factorization $g = a_1 \cdot \ldots \cdot a_s$ with $a_i \in A_i$.

Ex.: For a **subgroup chain**

$$G = G_0 > G_1 > \ldots > G_s = \{\mathrm{id}\}$$

choose A_i as left transversal of $G_{i-1} \bmod G_i$

The trouble with the instances...

MST$_1$ needs log. signatures Λ so that **factoring** w.r.t. Λ **needs** knowledge of **trapdoor**

$$\mathbb{Z}/|G|\mathbb{Z} \begin{smallmatrix}\longrightarrow\\ \dashleftarrow\end{smallmatrix} [A_1,\dots,A_s] \begin{smallmatrix}\longrightarrow\\ \cancel{\dashleftarrow}\end{smallmatrix} G \begin{smallmatrix}\longrightarrow\\ \dashleftarrow\end{smallmatrix} [B_1,\dots,B_t] \begin{smallmatrix}\longrightarrow\\ \dashleftarrow\end{smallmatrix} \mathbb{Z}/|G|\mathbb{Z}$$

$$n \xrightarrow{\text{mixed basis represent.}} (a_1,\dots,a_s) \xrightarrow{\text{multiply}} g \xrightarrow{\text{factor}} (b_1,\dots,b_t) \xrightarrow{\text{mixed basis represent.}} f(n)$$

... unclear how to generate these reliably (cf. key generation trouble w/ conjugacy & root problem in braid groups)

Symmetric cryptography & group theory

- For PGM, “a symmetric MST_1”, things look better: successful cryptanalysis lacking

- Tillich-Zémor hash function from CRYPTO 94 “structurally unbroken” for n prime:

1.) Fix $GF(2^n)=GF(2)[X]/(f(X))$

2.) For α a root of $f(X)$, set

$$B_0 := \begin{pmatrix} \alpha & 1 \\ 1 & 0 \end{pmatrix}, B_1 := \begin{pmatrix} \alpha & \alpha+1 \\ 1 & 1 \end{pmatrix} \in SL_2(GF(2^n))$$

3.) Hash value of $m \in \{0,1\}^*$: $H(m) := \prod_i B_i$

Getting constructive...

- Alternative to constructing complete group-based cryptographic scheme from scratch:

Provable reduction of, e.g., IND-CCA2 security to group-theoretical assumption
(➡ Cramer-Shoup-like construction).

- possibly no efficient instance known

+ you know what you need & get

Key establishment: what to expect?

Common requirements:

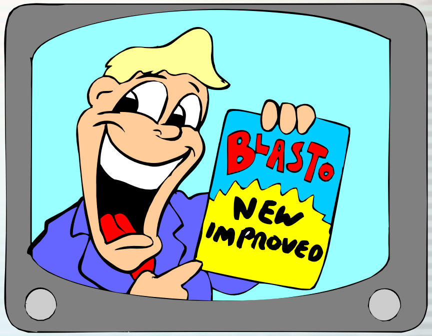
- key: uniformly at random chosen bitstring (e.g., to key a block or stream cipher)
- adversary cannot distinguish real key from uniform. at rand. chosen key space element
- concurrent protocol executions possible
- provide session identifier “naming” the key
- old session keys can be revealed
- forward security

Can we formalize this?

Security models for key establishment exist:

- successfully applied to “real life schemes”
- cover large class of attacks (but not “everything”, e.g., DoS typically ignored)
- compiler making passively secure protocol secure against active adversaries exists

... why not using the available tools when building on group theory?



Along the lines of Bellare, Bresson, ...

An approach to model group key establishment:

- ppt users $U_1, \ldots, U_r$ (r constant or polynomial)
- each user U_i runs several processes $\Pi_{i,j}$ (processes materialize concurrent executions)
- adversarial capabilities captured by oracles, incl. for the **passive** case:

  Execute($U_1,\ldots,U_r$) – get protocol transcript

  Reveal(U_i, j) – get session key & sid

More serious attacks ...

Note: already passive model allows multiple executions & compromised session keys

Additional oracles for active adversary:

Send(U_i, j, M) – “the adversary is the network”

Corrupt(U_i) – learn long term secret key (to address forward security)

... what is a “secure” key establishment?

Basic security requirement

Standard requirement concerns key secrecy:

Adversary queries Test(U_i, j) on **fresh** instance

Probability ½: receives correct secret key

Probability ½: uniformly at random chosen element from key space

... if s/he can distinguish non-negl. better than guessing, the scheme is not secure

Group-based key establishment revisited

- These requirements can be met efficiently.
- Can we base an efficient, say 2-round, group key establishment meeting such standards on group theory?

... being serious, we have to try.

Here: an attempt towards this goal (joint work with Benjamin Glas and Jens-Matthias Bohli)

Setting the scene (informal)

G: a group (more formally, a family $G = G(k)$...)
along with (stateless) ppt algorithms

`DomPar`:	chooses subgroup generators S
`SamAut`:	upon input S chooses $\phi \in \mathrm{Aut}(G)$
`SamSub`:	samples a word $x(S) \in \langle S \rangle$

Ex.: `DomPar` could fix cyclic group generator
`SamAut` could select exponent or inner aut.

Hardness assumption (informal)

Fix $r \in \mathbb{N}$ and $(G, \texttt{DomPar}, \texttt{SamAut}, \texttt{SamSub})$.

Given S, $(\phi_i(S), \phi_i(x))_{1 \leq i \leq r}$, no ppt algorithm recovers x with non-negl. probability, where

$$S \leftarrow \texttt{DomGen}(1^k)$$
$$x(S) \leftarrow \texttt{SamSub}(1^k, S)$$
$$\phi_i \leftarrow \texttt{SamAut}(1^k, S)$$

Ex.: polyn. time equival. to a CDH assumption

Note: independent of r

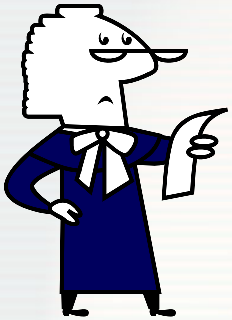
Does this assumption make sense?

- right now no specific non-abelian example; thinking about inner autom. looks tempting

  → more (group-theoretical) work needed

- polynomial time equivalence to “ordinary” CDH assumption gives us concrete instances

  → a concrete provably secure protocol



No “real non-abelian protocol”, but perhaps a helpful step.

Some technical comments

- proof uses random oracle model
- CMA-secure signature scheme used
- in general #parties must be constant; for CDH we can allow polynomial #parties
- some guarantees for honest players in the presence of malicious insiders
- forward security (long term keys only to sign)
- session identifier established within protocol
- “everybody does the same” (cf. Anshel et al.)

Protocol description

Principal U_i , process s_i, participants **U**, $|\mathbf{U}|=r$:

Round 1: Initialization $\text{pid}_{i,s_i}:=\mathbf{U}$, $\text{used}_{i,s_i}:=\text{true}$

Choose $\phi_{i,s_i} \leftarrow \texttt{SamAut}(1^k, S)$

$x_{i,s_i}(S) \leftarrow \texttt{SamSub}(1^k, S)$

Compute $m_{1,s_i}(U_i) :=((\phi_{i,s_i}(t))_{t\in S}, \mathrm{H}(x_{i,s_i}))$

Broadcast: $m_{1,s_i}(U_i)$

Means:over point-to-point connections

Protocol description (continued)

Round 2: Key Exchange

$\mathrm{sid}_{i,s_i} := \mathrm{H}(m_{1,s_1}(U_i), \ldots, m_{1,s_r}(U_r), \mathrm{pid}_{i,s_i})$

Compute and send the message

$$m_{2,s_i}(U_i, U_j) := (\phi_{j,s_j}(x_{i,s_i}), \mathrm{Sig}_i(\mathrm{sid}_{i,s_i}))$$

to each participant $U_j \in \mathrm{pid}_{i,s_i}$, $j \neq i$

... using the representation in terms of S

Efficiency drawback:

different message for each U_j , $j \neq i$

Round 2 (continued)

Key Generation Compute from $\phi_{i,s_i}(x_{j,s_j})$, the original x_{j,s_j} for all $j \neq i$ by applying the inverse of ϕ_{i,s_i}. Compute the common session key

$$K := \mathrm{H}(x_{1,s_1}, \ldots, x_{r,s_r}, \mathrm{pid}_{i,s_i})$$

Verification Check for all $U_j \in \mathrm{pid}_{i,s_i}$ if $\mathrm{Sig}_j(\mathrm{sid}_{j,s_j})$ is a valid signature for sid_{i,s_i} and if for x_{j,s_j} the received hash value $\mathrm{H}(x_{i,s_i})$ in $m_{1,s_j}(U_j)$ was correct.

If true, set $\mathrm{acc}_{i,s_i} := \mathrm{term}_{i,s_i} :=$ true, and $\mathrm{sk}_{i,s_i} := K$.

Else set $\mathrm{acc}_{i,s_i} :=$ false, $\mathrm{term}_{i,s_i} :=$ true.

Going on...

- Group theory and cryptography can have a fruitful exchange of ideas

  → proposals based on braid groups

- Precise security models help to avoid misunderstandings & “mathematically poor” attacks

More research is still needed to decide whether something “practical & non-abelian” is possible.