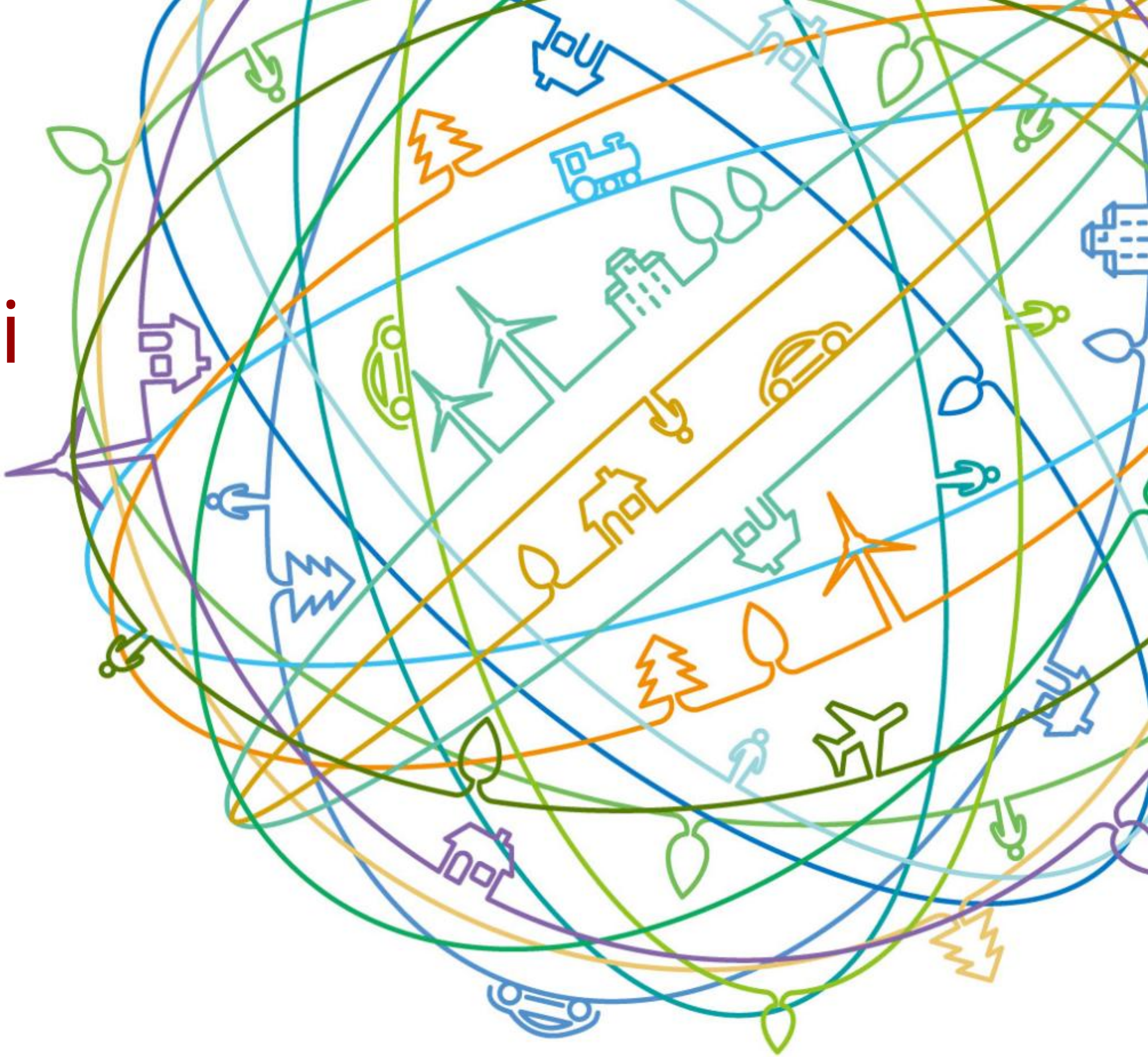
VTBPEKE: Verifier-based Two-Basis Password Exponential Key Exchange



IETF 101, London
March, 2018

Guilin Wang
(wang.guilin@Huawei.com)

Content

- **PAKE:** Terminology, Challenges, Existing Solutions
- **Our Proposals:** TBPEKE, VTBPEKE, Comparison, and Implementation
- **Candidate proposal for RFC 8125?**

PAKE: Terminology

- PAKE: Password-Authenticated Key Exchange, enable two parties to establish a shared cryptographically strong key over an insecure network using a short common secret as authentication means.
- Dictionary Attack: attackers guess users' passwords from a dictionary.
  - Online Dictionary Attack: try dictionary attack on line, which can be countered by limiting the times of trial in a given period.
  - Offline Dictionary Attack: Attackers do offline computations to recover users' passwords, after intercepting messages of some PAKE executions.
- Forward Security: Even if the password is later leaked, the privacy of a past communication is still guaranteed.
- Server Corruption: In normal PAKE, the compromising of the server may allow immediate leakage of all the passwords.
- VPAKE: Verifier PAKE for resisting server corruption, in which the server only stores a verifier of user's password, like the hash value of a password.

PAKE: Challenges and Our Work

- **Main challenge:** To design PAKE secure against offline dictionary attack.
- **Limitations in most of existing PAKE solutions:**
  - **Do not support forward security**, but this is essential to guarantee the privacy of a past communication.
  - **Security only proved in the multiplicative groups of finite fields,** which implies having to use large elements and then conduct huge communications and computations**.**

- We propose two PAKE protocols, which meet **forward security** and **works in any group.** http://www.di.ens.fr/users/pointche/Documents/Papers/2017_asiaccsB.pdf
  - **TBPEKE:** Two-Basis Password Exponential Key Exchange
  - **VTBPEKE:** a verifier-based variant of TBPEKE
- Both protocols **are proveably secure** under standard complexity assumptions.
- **Elliptic curves** can be used to implement the protocols, which leads to better efficiency, for both communication and computation.

PAKE: Existing Solutions

1) EKE (Encrypted Key Exchange)[BM92]:

$$
\begin{array}{lcl}
A^{\,pw} & & B^{\,pw} \\
x \xleftarrow{R} \mathbb{Z}_p^*, X \leftarrow g^x, X' \leftarrow \mathcal{E}_{pw}(X) & \xrightarrow{\quad A\|X' \quad} & X \leftarrow \mathcal{D}_{pw}(X') \\
Y \leftarrow \mathcal{D}_{pw}(Y') & \xleftarrow{\quad Y' \quad} & y \xleftarrow{R} \mathbb{Z}_p^*, Y \leftarrow g^y, Y' \leftarrow \mathcal{E}_{pw}(Y) \\
Z \leftarrow Y^x & & Z \leftarrow X^y \\
 & sk \leftarrow H(A\|B\|X\|Y\|Z) &
\end{array}
$$

Comments on EKE:

- The first PAKE protocol， proposed by Bellovin and Merrit.
- **Basic Idea**: **Password pw is used as a symmetric key to improve the security of DH key exchange.**
- **Security:** Security under random oracle model [BMP00] and under UC model [ACCP08], by the symmetric encryption is a ideal cipher)

5) SPEKE (Simple Password Exponential Key Exchange) [Jab96]:

$$
\begin{array}{lcl}
A^{\,pw} & & B^{\,pw} \\
\multicolumn{3}{c}{\mathbb{G} \text{ of prime order } p,\ G \text{ a hash function onto } \mathbb{G}, \text{ and } H \text{ a hash function onto } \{0,1\}^\ell} \\
\mathsf{accept} \leftarrow \mathsf{false} & & \mathsf{accept} \leftarrow \mathsf{false} \\
g \leftarrow H(pw), x \xleftarrow{R} \mathbb{Z}_p^*, X \leftarrow g^x & \xrightarrow{\quad A\|X \quad} & \\
 & \xleftarrow{\quad Y \quad} & g \leftarrow H(pw), y \xleftarrow{R} \mathbb{Z}_p^*, Y \leftarrow g^y \\
\mathsf{accept} \leftarrow \mathsf{true} & & \mathsf{accept} \leftarrow \mathsf{true} \\
Z \leftarrow Y^x & & Z \leftarrow X^y \\
 & sk \leftarrow H(A\|B\|g\|X\|Y\|Z) &
\end{array}
$$

Comments on SPEKE:

- Proposed by David Jablon [Jab96a, Jab96b]
- **Basic Idea:** *g=H(pw)*, i.e., *pw* is used to generate the generator.
- **Security**: Provable security in the BPR model [Mac01] under the CDH assumption. But the proof applies only to a multiplicative sub-group of finite fields Z_p*, not for ECC groups.
- **Efficiency:** Due to the above reason, big group size, not efficient in both communication and computation.

Page 5

Our Proposals: TBPEKE

- **TBPEKE** is an improvement of SPEKE to make it more efficient and secure.
- Security proof applies to ECC and also a group of finite fields.
- **Basic Idea:** **g=H(pw) is used in SPEKE, and we here define g=UV^{pw}.**
- **Two bases (U and V) used here, inspired by SPAKE.**

$$
\begin{array}{rcl}
A \; pw & & B \; pw \\
\multicolumn{3}{c}{\mathbb{G} \text{ of prime order } p,\ U, V \overset{R}{\leftarrow} \mathbb{G}, \text{ and } H \text{ a hash function onto } \{0,1\}^{\ell}} \\
\mathsf{accept} \leftarrow \mathsf{false} & & \mathsf{accept} \leftarrow \mathsf{false} \\
g \leftarrow U \cdot V^{pw}, x \overset{R}{\leftarrow} \mathbb{Z}_p^*, X \leftarrow g^x & \xrightarrow{\quad A \| X \quad} & \\
& \xleftarrow{\quad Y \quad} & g \leftarrow U \cdot V^{pw}, y \overset{R}{\leftarrow} \mathbb{Z}_p^*, Y \leftarrow g^y \\
\mathsf{accept} \leftarrow \mathsf{true} & & \mathsf{accept} \leftarrow \mathsf{true} \\
Z \leftarrow Y^x & & Z \leftarrow X^y \\
\multicolumn{3}{c}{sk \leftarrow H(A \| B \| g \| X \| Y \| Z)}
\end{array}
$$

Page 6

Our Proposals: VTBPEKE

- A client (A) saves *pw*, while the server (B) saves a salt *s* and the verifier $V^{H(s,pw)}$.
- **Basic Idea:** 1) A needs to prove its knowledge of *H*(*s*, *pw*)., for preventing an attack to impersonate A after getting the verifier. 2) The proof response ρ is encrypted for prevent off-line dictionary attack to guess *pw*.

$$A^{\,pw} \qquad\qquad B^{\,s,\mathcal{V}} \quad \text{where } \mathcal{P} \leftarrow \mathcal{H}_p(s, pw) \text{ and } \mathcal{V} \leftarrow V^{\mathcal{P}}$$

Global parameters: $\mathbb{G}$ of prime order p, $U, V \overset{R}{\leftarrow} \mathbb{G}$, $\mathcal{H}_p$ a hash function onto $\mathbb{Z}_p$, and $\mathcal{H}$ onto $\{0,1\}^k$

A		B
accept ← false		accept ← false
$\alpha \overset{R}{\leftarrow} \mathbb{Z}_p^*, R \leftarrow V^{\alpha}, R' \leftarrow \mathcal{H}(11 \Vert R)$	$\xrightarrow{A \Vert R'}$	$g \leftarrow U \cdot \mathcal{V}, y \overset{R}{\leftarrow} \mathbb{Z}_p^*, Y \leftarrow g^y$
$\mathcal{P} \leftarrow \mathcal{H}_p(s, pw), \mathcal{V} \leftarrow V^{\mathcal{P}}$	$\xleftarrow{s, \varepsilon, Y}$	$\varepsilon \overset{R}{\leftarrow} \{0,1\}^k$
$g \leftarrow U \cdot \mathcal{V}, x \overset{R}{\leftarrow} \mathbb{Z}_p^*, X \leftarrow g^x, Z \leftarrow Y^x$		
$\rho \leftarrow \alpha + \varepsilon \cdot \mathcal{P} \bmod p, \sigma \leftarrow \mathcal{E}_{ek}(\rho)$	$\xrightarrow{X, \sigma}$	$Z \leftarrow X^y, \rho \leftarrow \mathcal{D}_{ek}(\sigma)$
		$R \leftarrow V^{\rho}\mathcal{V}^{-\varepsilon}, R' \overset{?}{=} \mathcal{H}(11 \Vert R)$
accept ← true		accept ← true

$$ek \leftarrow \mathcal{H}(01 \Vert A \Vert B \Vert g \Vert X \Vert Y \Vert Z \Vert R' \Vert s \Vert \varepsilon)$$
$$sk \leftarrow \mathcal{H}(00 \Vert A \Vert B \Vert g \Vert X \Vert Y \Vert Z \Vert R' \Vert s \Vert \varepsilon \Vert \sigma)$$

Page 7

Security Proofs

- **Security Model:**
  - Random oracle+Real-or-Random Game
- **Proofs in several cases:**
  - If Forward Security is required?
  - Password dictionary is considered in 3 cases (Large, Medium, Small)
  - Under generic model
- **Hard problem assumptions:**
  - CDH, Dlin, SDH (new problem introdcued)
  - SDH is not easier than DLin [Theorem 1]
  - If Forward Security not required: CDH and SDH
  - If Forward Security required: GCDH and GSDH (G for Gap)

Example: When using bilinear and forward security is required, in this case GDLin and GCDH are actually Dlin and CDH 。 There is the security result:

Bilinear Settings

Note however that this reduction is really meaningful when the DDH oracle is efficient, which requires a bilinear map. Then, in such a case, the security simply relies on the DLin and the CDH assumptions:

Theorem 6. *In the bilinear setting, under the DLin and CDH assumptions, the TBPEKE is a forward-secure PAKE. More precisely, the best advantage an adversary can get in the Real-or-Random security game (see Figure 1) is bounded by*

$$\mathsf{Adv}(t) \leq \frac{q_s}{N} + N^2 \times \mathsf{Adv}^{\mathsf{dlin}}(t) + \mathsf{Succ}^{\mathsf{cdh}}(t) + \frac{q_e q_s}{p^2}, \text{ if the dictionary is small, or}$$

$$\mathsf{Adv}(t) \leq q_s \times \sqrt{10 N_C \times \mathsf{Adv}^{\mathsf{dlin}}(t)} + \mathsf{Succ}^{\mathsf{cdh}}(t) + \frac{q_e q_s}{p^2}, \text{ if the dictionary is large.}$$

Page 8

Comparison

Scheme	Communication (Both Sides)	Computation (Both Sides)	Forward Secrecy	Security Model	Assumptions	Limitations
PAKE						
EKE [12]	$1\mathbb{G}$ / $1\mathbb{G}$	2E / 2E	Yes	ICM	CDH [19]	
SPAKE [6]	$1\mathbb{G}$ / $1\mathbb{G}$	2E+2sE / 2E+2sE	No	ROM	CDH	
SPAKE2 [6]	$1\mathbb{G}$ / $1\mathbb{G}$	2E+2sE / 2E+2sE	No	ROM	CDH	
SPEKE [33]	$1\mathbb{G}$ / $1\mathbb{G}$	2E / 2E	No	ROM	CDH	in $\mathbb{Z}_p^*$ only
SAE [32]	$2\mathbb{G}$ / $2\mathbb{G}$	3E / 3E	No	ROM	CDH [42]	in $\mathbb{Z}_p^*$ only
SRP [50]	$3\mathbb{G}$ / $3\mathbb{G}$	2E / 3E	No proof			in $\mathbb{Z}_p^*$ only
GL-SPOKE [2]	$4\mathbb{G}$ / $3\mathbb{G}$	10E /10E	Yes	Standard	DDH	
GK-SOPKE [2]	$2\mathbb{G}$ / $4\mathbb{G}$	8E / 9E	Yes	Standard	DDH	
TBPEKE	$1\mathbb{G}$ / $1\mathbb{G}$	2E+1sE / 2E+1sE	**Yes**	ROM	GSDH	
Verifier-based PAKE						
SPAKE2+ [23]	$1\mathbb{G}$ / $1\mathbb{G}$	5E / 5E	No	ROM	CDH	
AugPAKE [45]	$1\mathbb{G} + k$ / $1\mathbb{G} + k$	2E / 3E	No	ROM	Strong DH [48]	
VTBPEKE	$1\mathbb{G} + k + \|p\|$ / $1\mathbb{G} + k$	4E / 4E	**Yes**	ROM	GSDH	

Recommended Parameters

64 bits	4-Digit	8-Char	32-Char
Without FS	347	401	514
With FS	218*	272*	386*

112 bits	4-Digit	8-Char
Without FS	587	641
With FS	362*	416*

80 bits	4-Digit	8-Char	40-Char
Without FS	427	441	642
With FS	266*	320*	482*

128 bits	4-Digit	8-Char
Without FS	667	721
With FS	410*	464*

* in pairing-friendly elliptic curves

Fig. 6. Parameters: Bit-length of the order of the groups

- **4 security levels are given**
- For each security level, 3 case of password size are given.
- For each combination of security level and password size, the required bit length of ECC element is given.

Page 10

Implementation

OPENSSL Based Implementation

Implementation Environments:

- CPU1 ： Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz
- CPU2 ： Intel(R) Xeon(R) CPU E3-1230 v3 @ 3.30GHz
- NIST P-256 and P-521 curves only
- Each CPU runs the computations by both parties of TBPEKE, and no network communication.

Testing Results: The time used for running the TBPEKE protocol 3000 times

- CPU1， P-256 curve: 6.415319 s
- CPU1， P-521 curve: 29.851816 s
- CPU2， P-256 curve: 4.728513 s
- CPU2， P-521 curve: 21.028553 s

Average per time running time: < 10ms

Page 11

A Candidate Proposal for RFC 8125?

RFC 8125: Requirements for PAKE protocols (**https://tools.ietf.org/html/rfc8125**)

- This document reviews different types of PAKE schemes.
- It presents requirements and gives recommendations to designers of new schemes.

8 Requirements for PAKE, given by RFC 8125：

- **REQ1:** A PAKE scheme MUST clearly state its features regarding balanced/augmented versions.
- **REQ2:** A PAKE scheme SHOULD come with a security proof and clearly state its assumptions and models.
- **REQ3:** The authors SHOULD show how to protect their PAKE scheme implementation in hostile environments, particularly, how to implement their scheme in constant time to prevent timing attacks.
- **REQ4:** If the PAKE scheme is intended to be used with ECC, the authors SHOULD discuss their requirements for a potential mapping or define a mapping to be used with the scheme.
- **REQ5:** The authors of a PAKE scheme MAY discuss its design choice with regard to performance, i.e., its optimization goals.
- **REQ6:** The authors of a scheme MAY discuss variations of their scheme that allow the use in special application scenarios. In particular, techniques that facilitate long-term (public) key agreement are encouraged.
- **REQ7:** Authors of a scheme MAY discuss special ideas and solutions on privacy protection of its users.
- **REQ8:** The authors MUST follow the IRTF IPR policy https://irtf.org/ipr.

Thank you

www.huawei.com