# On Small Subgroup Non-confinement Attack

Feng Hao
*Thales E-Security, Cambridge, UK*
*Feng.Hao@thales-esecurity.com*

***Abstract*—The small subgroup confinement attack works by confining cryptographic operations within a small subgroup, in which exhaustive search is feasible. This attack is overt and hence can be easily thwarted by adding a public key validation: verifying the received group element has proper order. In this paper, we present a different aspect of the small subgroup attack. Sometimes, the fact that an operation does not fall into the small subgroup confinement may provide an oracle to an attacker, leaking partial information about the long-term secrets. This attack is subtle and reflects structural weakness of a protocol; the question of whether the protocol has a public key validation is completely irrelevant. As a concrete example, we show how this attack works on the Secure Remote Password (SRP-6) protocol.**

***Keywords*-password authenticated key exchange, secure communication, Secure Remote Password protocol**

## I. Introduction

The small subgroup confinement attack is one common attack against discrete logarithm based key agreement protocols [3], [4]. It exploits the structure of the group $G$ where key agreement takes place. One choice of such a group is $Z_p^*$ where $p$ is a large prime. The order of this group is a composite, so there exist subgroups. Say $G_w$ is one small subgroup of primer order $w$, then $w|p-1$. Suppose $g$ is a non-identity element in $G_w$, then $g^x$ for $x \in Z_p$ will also lie in the same subgroup. This can potentially cause problem if $w$ is small: an adversary can then exhaustively search all elements in the subgroup.

One attacking scenario is to confine the session key derived from a key agreement protocol to a small set. In the original description of the Diffie-Hellman protocol [16], the key agreement operates in the whole cyclic group $Z_p^*$. Alice and Bob select random secrets $x$ and $y$ respectively ($x, y \in_R [1, p-1]$), and exchange ephemeral public keys $A = \alpha^x$ and $B = \alpha^y$ where $\alpha$ is a primitive root modulo $p$. In the end, Alice and Bob can compute a common key $K = \alpha^{xy}$, which a passive attack cannot. An active attacker can however confine $K$ to a small subgroup $G_w$ as follows. He intercepts $A$, $B$ and replaces $A$ with $A^{(p-1)/w}$, $B$ with $B^{(p-1)/w}$. The common key computed by Alice and Bob will be $K = \alpha^{xy(p-1)/w}$. Because $K$ is an element of the small subgroup, the attacker can find out $K$ by exhaustive search.

The above attack motivates moving the operation of key agreement from the whole group $Z_p^*$ to a large subgroup. Since Pollard's rho method can compute a logarithm in a subgroup of prime order $q$ in time $O(\sqrt{q})$, the size of the subgroup must be at least twice the intended security level [13]. In other words, for 80-bit security, $q$ should be at least 160-bit. The Diffie-Hellman protocol can be modified accordingly: Alice and Bob just need to change the $\alpha$ to a generator $\beta$ of the intended subgroup, which for example can be computed as $\beta = \alpha^{(p-1)/q}$.

In addition, it is important for Alice and Bob to validate the received public keys. This can be done by verifying $A, B \in (1, p-1)$, and also $A^q, B^q = 1$. However, this verification adds burden to the end user's computation as the exponentiation is an expensive operation. An alternative is to choose secure group parameters such that $(p-1)/2$ contains no small factors (say all larger than 160-bits) and in the extreme case $(p-1)/2$ is a prime itself. This approach will make the generation of group parameters substantially more expensive (which is however not a big problem if the parameters are computed only once for a community of users). Choosing secure parameters may mitigate the effects of certain attacks, but it does not fundamentally change the problem. In any event there will always be at least one small subgroup, the one containing only two elements $\{1, p-1\}$. Hence, proper validation is still needed to ensure not falling into this subgroup.

With an appropriate public key validation in place, it appears that small subgroup attacks can be prevented. This seems the case for all the reported small subgroup attacks so far [3]–[5].

In this paper, we demonstrate a different (or rather paradoxical) aspect of the small subgroup attack: attackers may exploit the small subgroup "non-confinement". While some cryptographic protocols were carefully designed to avoid falling into small subgroups "traps", the fact that they did not fall into the "traps" provides an oracle to the attacker, leaking partial information about the long-term secrets. This attack is especially concerning when the long-term secrets are low-entropy passwords. In the following sections, we will explain how this attack affects the Secure Remote Password (SRP) protocol.

## II. SRP-6 protocol

The Secure Remote Password (SRP) protocol was first proposed by Wu in 1998 [1], [2]. It aims to address the Password Authenticated Key Exchange (PAKE) problem – namely, how to bootstrap a high-entropy session key

based on a low-entropy shared password without requiring a Public Key Infrastructure. The SRP protocol has been deployed in many practical applications and is currently being standardized by the IEEE P1363.2 working group [8].

The development of the SRP technique follows a heuristic approach. The protocol bases its security upon gradual improvements. When new attacks were discovered, the protocol was patched accordingly.

Over the years, the protocol has been revised several times. The SRP-3 [1] is the initial submission to the IEEE P1363.2 Standardization committee [8]. But it was later found containing various weaknesses, among which the most serious one is the subjection to a two-for-one guess attack (discovered by D. Bleichenbacher in 2000) [2]. An active attacker was able to exploit a design flaw in SRP-3 to test two passwords in one attempt. The protocol was revised to address that attack and after several attempted changes, the final version called SRP-6 was submitted to IEEE P1363.2 [2].

After patching the two-for-one attack, the SRP author claims that SRP-6 has achieved the theoretical limit in permitting exactly one password guess per protocol execution [2]. However, this theoretical claim was made without any theoretical proof. We will show a counterexample to indicate otherwise.

First, we explain how SRP-6 works (Figure 1). The protocol operates in a group defined by a safe prime $N = 2 \times p + 1$ where $p$ is also a prime. All values in Figure 1 are computed modulo $N$.

The client and server bootstrap their trust relationship based on a common password $P$. Let $s$ be a random salt and $I$ be the user identity. In SRP-6, the server does not store $P$; instead it applies a one-way hash function $H$ to computes $x = H(s, I, P)$ and only stores a *verifier* $v = g^x$ where $g$ is a primitive root modulo $N$. Details of each step in the protocol are explained below:

1) The client sends his identity $I$ to the server.
2) The server replies with the salt $s$ after looking up $I$ in the database.
3) The client chooses a random number $a$, $1 < a < N$, and sends the ephemeral public key $A = g^a$ to the server.
4) After verifying $A \neq 0$, the server chooses a random number $b$, $1 < b < N$, and sends $B = 3v + g^b$ to the client.
5) After verifying $B \neq 0$, the client computes $M_1 = H(A, B, S)$ where $S = (B - 3g^x)^{a+H(A,B)\cdot x}$ and sends $M_1$ to the server.
6) After verifying $M_1$, the server computes $M_2 = H(A, M_1, S)$, and send $M_2$ to the client.
7) Finally, both sides can compute a common session key $k = H(S)$.

## III. Attack

In SRP-6, the modulus $N$ is defined to be a safe prime $N = 2 \times p + 1$ where $p$ is also a prime. The use of a safe prime is a common choice in key exchange protocols, for example SPEKE [11]. The main rationale is to address a potential attack where an active attacker may exploit the existence of small-order subgroups to confine the exponentiation operation within a small subgroup. One countermeasure is to perform a public key validation: checking the received group element has the proper order. It requires one exponentiation to do that. Another means is to use a safe prime. Then, there will be only one small subgroup that contains two elements $\{1,\ N-1\}$. Checking that an element does not fall within this small subgroup is very easy.

However, the use of a safe prime is not a panacea. It still depends on how the elements in the protocol interact with each other. In particular, we observe that in Step 4 (Figure 1), it is unclear which subgroup that the base $B-3g^x$ generates. Normally, for cipher algorithms built upon the intractability of discrete logarithm problems, such as the DSA and Diffie-Hellman key exchange, they operate in a subgroup (of prime order) consistently [13]. But in SRP-6, the base $B-3g^x$ may have order 2, or $p$, or $2p$. This ambiguity can be exploited by an active attacker, as we explain below.

Let us consider the case where the client holds a secret $x = H(s, I, P)$ and the server (attacker) does not have the correct verifier ($v = g^x$). So the attacker takes a random guess of the password $P'$, and computes $x' = H(s, I, P')$, $v' = g^{x'}$. Figure 2 demonstrates the attack. Details of each step are explained as follows.

1) The client sends his identity $I$ to the server.
2) The server sends to the client the salt $s$ (which the attacker eavesdropped in the past).
3) The client chooses a random number $a$, $1 < a < N$, and sends the ephemeral public key $A = g^a$ to the server.
4) The server chooses a random guess of the password $P'$, computes $v' = g^{H(s,I,P')}$ and sends $B' = 3v'$ to the client.
5) After verifying $B' \neq 0$, the client computes $M_1 = H(A, B', S)$ and sends $M_1$ to the server.

Subsequently, the server (attacker) goes off-line and does the following computation. If he finds $M_1 = H(A, B', 0)$, that is $S = (B' - 3g^x)^{a+ux} = 0$, then he had guessed the right password. What if he guessed wrongly? The theoretical limit of a PAKE protocol is to limit the attacker to learn nothing more than $P \neq P'$ (i.e., the zero-knowledge property of the protocol). Though SRP-6 claims to provide the strict zero-knowledge verification of the password, the following analysis indicates otherwise.

With overwhelming probability, $S \neq 1$ or $N - 1$ because the value $S$ is random over $[1, N - 1]$. (If $S$ happens to be 1 or $N - 1$, the attacker simply needs to try a different

| | Client | | Server |
|---|---|---|---|
| 1. | | $\xrightarrow{I}$ | (look up s, v) |
| 2. | $x = H(s, I, P)$ | $\xleftarrow{s}$ | |
| 3. | $A = g^a$ | $\xrightarrow{A}$ | $B = 3v + g^b$, $u = H(A, B)$ |
| 4. | $u = H(A, B)$, $S = (B - 3g^x)^{a+ux}$ | $\xleftarrow{B}$ | $S = (Av^u)^b$ |
| 5. | $M_1 = H(A, B, S)$ | $\xrightarrow{M_1}$ | (verify $M_1$) |
| 6. | (verify $M_2$) | $\xleftarrow{M_2}$ | $M_2 = H(A, M_1, S)$ |
| 7. | $K = H(S)$ | | $K = H(S)$ |

Figure 1. The SRP-6 protocol

| | Client | | Server (attacker) |
|---|---|---|---|
| 1. | | $\xrightarrow{I}$ | (Look up eavesdropped $s$) |
| 2. | $x = H(s, I, P)$ | $\xleftarrow{s}$ | |
| 3. | $A = g^a$ | $\xrightarrow{A}$ | (Choose $x'$), $B' = 3v' = 3g^{x'}$ |
| 4. | $u = H(A, B')$, $S = (B' - 3g^x)^{a+ux}$ | $\xleftarrow{B'}$ | |
| 5. | $M_1 = H(A, B', S)$ | $\xrightarrow{M_1}$ | if $M_1 = H(A, B', 0)$, obtain $x' = x$; |
| | | | else if $M_1 = H(A, B', N-1)$, stop; |
| | | | else if $M_1 = H(A, B', 1)$, stop; |
| | | | else: $3v' - 3v \neq 0$ |
| | | | $3v' - 3v \neq 1$ |
| | | | $3v' - 3v \neq N - 1$ |

Figure 2. An active attack on SRP-6

password next time such that $S \neq 1$ or $N-1$.) The attacker can verify this by checking $M_1 \neq H(A, B', N-1)$ and $M_1 \neq H(A, B', 1)$. Thus, the attacker concludes that $S = (B' - 3g^x)^{a+ux}$ is not one of the small subgroup elements $\{1, N-1\}$, so it must be case that the base $B' - 3g^x \neq 1$ and $B' - 3g^x \neq N - 1$. (Note the $N-1$ is the generator of the small subgroup). Hence, the attacker learns additional information $3v' - 3v \neq 1$ and $3v' - 3v \neq N - 1$. Based on this, he rules out two more verifiers, which correspond to two passwords. By the SRP-6 protocol definition, the password $P$ and the verifier $v$ (as well as all the other values in Figure 1) are defined modulo $N$ [2]. Therefore, the two filtered verifiers are perfectly legitimate values by definition.

The attack can be slightly more complicated. In Step 4 of Figure 2, the attacker could send $B' = 3v' + g^b$ where $b$ is an arbitrary value of the attacker's choice. After Step 5, the attacker could immediately filter out three $v$ values based on $3v'-3v+g^b \neq 0$, $3v'-3v+g^b \neq 1$ and $3v'-3v+g^b \neq N-1$. However, this time, the attacker has to follow through the rest of the steps in the SRP-6 in order to learn whether $v = v'$ (that is whether $P = P'$). In summary, if the attacker guessed the password wrongly, he can filter out four verifies in one go, which correspond to four different $P$ values.

This attack may appear counterintuitive to protocol designers. Normally, a common goal in the protocol design is to prevent falling into small-order subgroups. However, in the case of SRP-6, the fact that the protocol does not fall into a small-order subgroup gives away information, allowing an active attacker to test multiple passwords in one attempt. Given the existing structural design of the SRP-6 protocol, there seem no easy ways to fix this issue .

We need to stress that this attack is subtle. We think it is unlikely to pose serious threat to the practical security of the SRP-6. The choice of a safe prime $N$ in SRP-6 has significantly mitigated the practical effect of the attack. If SRP-6 operates in a different group (with more than one small subgroups), the effect can be much worse. In any case, the theory of the attack remains exactly the same, which indicates a structural weakness of the SRP-6 protocol design.

However, this attack is still significant in some aspects. First, it shows the danger of making a theoretical claim in a security protocol without any theoretical proof. The SRP-6 author claims that the protocol has achieved the theoretical limit of the best on-line dictionary attack resistance [2]. The above attack provides a counterexample to suggest otherwise. This helps gain better understanding of the SRP-6 construction. Second, it cautions protocol designers on the other side of the small-subgroup attack: besides the confinement, an attacker may also exploit the non-confinement of a cryptographic operation. The more conventional approach is to design an algorithm to operate in a single prime-order subgroup unambiguously and consistently (e.g., DSA, Schnorr Signature, SPEKE [11] and J-PAKE [17] etc) rather than hop among subgroups of different orders (e.g., SRP-

6 [2]). The former is of course free from both the small-subgroup confinement and non-confinement attacks.

## IV. Conclusion

In this paper, we describe a paradoxical side of the small subgroup attack. While a protocol is designed to avoid falling into the small subgroup, sometimes the fact of not falling into the small subgroup may leak information. This happens when the protocol mixes up operations in subgroups of different orders. As a concrete example, we demonstrate how this attack works on the SRP-6 protocol. While the SRP-6 scheme claims that it has achieved the theoretical zero-knowledge verification of the password, this attack provides a counterexample to suggest otherwise.

## References


[1] T. Wu, "The Secure Remote Password Protocol," Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium, San Diego, CA, pp. 97-111, Mar 1998.

[2] T. Wu, "SRP-6: Improvements and Refinements to the Secure Remote Password Protocol," Submission to the IEEE P1363 Working Group, Oct 2002.

[3] C.H. Lim and P.J. Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup", Crypto '97, LNCS 1295, pp. 249-263, 1997.

[4] D. Brown, A. Menezes, "A Small Subgroup Attack on a Key Agreement Protocol of Arazi," Bulletin of the ICA, No. 37, pp. 45-50, 2003.

[5] C. Boyd, A. Mathuria, "*Protocols for authentication and key establishment*," Springer-Verlag, 2003.

[6] D. Taylor, T. Wu, N. Mavrogiannopoulos, T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication," RFC 5054, Nov 2007. RFC5054 http://tools.ietf.org/html/rfc5054

[7] SRP Protocol Design from the official SRP website: http://srp.stanford.edu/

[8] IEEE P1363 Working Group, "Draft standard for Specifications for Password-based Public Key Cryptographic Technique," IEEE P1363.2/D26, Sep 2006. P1363.2 http://grouper.ieee.org/groups/1363/

[9] S. Bellovin and M. Merritt, "Encrypted Key Exchange: password-based protocols secure against dictionary attacks," Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1992.

[10] B. Jaspan, "Dual-workfactor Encrypted Key Exchange: efficiently preventing password chaining and dictionary attacks," Proceedings of the Sixth Annual USENIX Security Conference, pp. 43-50, July 1996.

[11] D. Jablon, "Strong password-only authenticated key exchange," *ACM Computer Communications Review*, Vol. 26, No. 5, pp. 5-26, October 1996.

[12] Muxiang Zhang, "Analysis of the SPEKE password-authenticated key exchange protocol," *IEEE Communications Letters*, Vol. 8, No. 1, pp. 63-65, January 2004.

[13] D. Stinson, *Cryptography: theory and practice*, Third Edition, Chapman and Hall/CRC, 2006.

[14] V. Boyko, P. MacKenzie, and S. Patel, "Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman," Eurocrypt 2000, NCS 1807, pp. 156-171, 2000.

[15] "Password-authenticated key exchange (PAK) protocol," ITU-T Recommendation X.1035, Feb 2007. http://www.itu.int/rec/T-REC-X.1035/en

[16] W. Diffie, M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644-654, 1976.

[17] F. Hao, P. Ryan, "Password authenticated key exchange by juggling," the 16th International Workshop on Security Protocols, SPW'08, Cambridge, UK, May 2008.