Summary of AMP (Authentication and key agreement via Memorable Passwords)

Taekyoung Kwon

Sejong University, Seoul 143-747, Korea
tkwon@sejong.ac.kr

Abstract. This document summarizes the so-called AMP protocols.

1 Introduction

Due to the low entropy of human-memorable passwords, it is not easy to conduct password authenticated key agreement in a secure manner. Since a pioneering method was introduced for resisting password guessing attacks [14], there has been a great deal of related work for password authenticated key agreement on the framework of Diffie-Hellman [6], for example, EKE [2, 3] and so on [20, 22]. Readers are referred to [21] for complete references. Among them, SPEKE [9], SRP [26], PAK [16], and AMP [12] are being discussed by the IEEE P1363 standards working group for standardization on password-based public key cryptographic techniques [8]. Compared to the typical authenticated key agreement, the password-based schemes are totally expensive, specifically in the augmented model which could resist server compromise. From the theoretical perspective, several methods that are much more expensive, have been presented [7, 10]. Also the practical protocols have been proposed [15, 4], for example, PAK and SPEKE are known as the most acceptable 'three-pass' protocols to the standard body. From the practical perspective, AMP and SRP are known as the most efficient 'four-pass' protocols. In this document, we summarize the so-called AMP protocols and compare them with the related protocols.

The main contribution is the four-pass protocols (AMP) but the three-pass protocols (TP-AMP) which are combinations of AMP and PAK, are also described in this document.

1.1 This Document

This document is in draft and describes the following.

- Summary of AMP protocols
  - Four-pass protocols: AMP (Figure 1), AMP2 (Figure 2), AMP3 (Figure 3)
  - Three-pass protocols: TP-AMP (Figure 4), TP-AMP2 (Figure 5)
- Simple efficiency analysis
  - Four-pass protocols (Tables 1, 2 and 3)
  - Three-pass protocols (Tables 4 and 5)
- Miscellaneous items
  - Many-for-many guessing (Figure 10)
  - Related protocols: SRP5 (Figure 7), SRP6 (Figure 8), and PAK-Z (Figure 9)

1.2 Preliminaries

A client and a server should agree on algebraic parameters. Let κ and ℓ denote security parameters. Notice that κ is regarded as the general parameter for hash functions and secret keys (say 160 bits), while ℓ is thought of as the special parameter for public keys (say 1024 or 2048 bits). Let q of the length at least κ and p of the length ℓ be respectively primes such that $p = rq + 1$ for some value r co-prime to q. Let g be a generator of G_q where G_q is a q-order subgroup of a multiplicative group Z_p^*. Let us often omit " mod p" from the expressions that are contextually obvious in Z_p^*. We recommend to use a *secure prime* such that each factor of $\frac{r}{2}$ is of the size at least κ as discussed in [12, 13]. Let user remember one's *id* and password π, and work on a machine A where A can be thought of as an IP address. Similarly server may store a user profile on a machine B. Let $\{0,1\}^*$ denote the set of finite binary strings and $\{0,1\}^n$ the set of binary strings of length n. Then we could have random oracles denoted by $h_i : \{0,1\}^* \rightarrow \{0,1\}^n$ and instantiated by strong one-way hash functions. The size of n and the specific design of them are depends upon the original work [4, 12, 15, 16]. In this document, the size of n will be manipulated flexibly with regard to each scheme. For convenience we specify two functions with regard to the size of their image such that $h_i : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ and $H_i : \{0,1\}^* \rightarrow \{0,1\}^\ell$. Let $\text{ACCEPTABLE}_i()$ denote a function which may return true if its pre-image satisfies the given security properties. For more details, readers are referred to the previous work [4, 8, 12, 15, 16, 26, 27]. Readers are regarded as being familiar with the related protocol description.

2 AMP Protocols

AMP stands for the Authentication and key agreement via Memorable Passwords [12], and was contributed to the IEEE P1363.2 standard work. On standardization, the protocol was improved in order for resisting the newly found attack called two-for-one guessing and for minimizing the necessary computation. Also a three-pass derivative called TP-AMP (Three-Pass AMP) was proposed. We describe those AMP protocols briefly.

2.1 Four-Pass AMP Protocol

AMP was basically designed as a four-pass protocol in the augmented model [12]. Former protocols such as GXY [11] and SRP [26] were its motivation. Figure 1 depicts the AMP protocol that was refined in 2001 for resisting two-for-one guessing. In fact, the AMP protocol was replaced by AMP+ of the original work [12].

In the refined AMP, raising m to e_1 was necessary for resisting the two-for-one guessing. For example, an adversary should find out e_1 such that $e_1 = h_1(g^{-\frac{u'}{e_1}})$ for two-for-one guessing and its probability is negligible due to the property of the strong one-way hash function $h_1()$. So the parties should agree on the secret value, $g^{(x+e_2)y} = g^{xy}(g^y)^{e_2}$. Note the length of the value e_2 could be 40 bits, while e_1 is not. The AMP protocol is comparable with the SRP protocols and specifically SRP5 which is depicted in Figure 7. Also the AMP protocol can be constructed in a slightly different way for more efficiency as depicted in Figure 2. In AMP2, the values e_1 and e_2 are unified so that w could be computed before receiving μ.

Another lightweight version of AMP is depicted in Figure 3 and is called AMP3. It was devised to reduce the amount of computation from the server, so that a new function, $\textsc{acceptable}_2()$, is defined to return false if its pre-image is confined to a tiny subgroup. In spite that $\textsc{acceptable}_2()$ needs new constraints on checking small order elements, its computation can be minimized by choosing a specific prime such as a secure prime or a safe prime and the server could perform one $\bmod p$ exponentiation for μ. In AMP3, if $\textsc{acceptable}_2(c)$ returns false, the server may set μ as g^y that is a random-looking element in G_q. Otherwise, it sets μ as c^y. Also after sending μ, the server could check and abort if mg is confined to a tiny subgroup by using the same function. So the parties should agree on the secret value, $g^{(x+1)y} = g^{xy}(g^y)$. AMP3 is comparable with SRP6 which must use a safe prime.

2.2 Three-Pass AMP Protocol

TP-AMP is derived from AMP and PAK with regard to computational efficiency of three-pass protocols in the augmented model. On constructing TP-AMP, we can consider all AMP protocols for a combination with PAK. So we can call it AMP-PAK. Figure 4 depicts TP-AMP based on AMP2 and is comparable with PAK-Z with Y instance which is depicted in Figrue 9. Note that a client follows the procedure of PAK for computing out m, while it does that of AMP for computing α. Also the server may compute μ and β in the same way as AMP but responds with them as defined in PAK. Since no one can control the discrete logarithm of $m\gamma''$ if γ was not correct in m and $\gamma'' \neq \gamma^{-1}$, the server can respond both with μ and k_1 before receiving k_2, say in three passes. This must simplify the three pass protocol in the augmented model.

TP-AMP2, as depicted in Figure 5, is another derivative of TP-AMP in which AMP and PAK are merged in a slightly different way that μ removes $(m')^y$ in its composition and instead the hash value e raises g^y in β. Note that e was necessary for resisting server compromise in TP-AMP2. If e of TP-AMP2 is computed in the same way as e_2 of AMP, its length can be reduced into 40 bits only. When we consider the formal security of AMP protocols, we may see that TP-AMP2 is based upon the CDH (Computational Diffie-Hellman) problem, while the other AMP protocols is on the DDH (Decision Diffie-Hellman) problem due to the intrinsic nature of μ.

2.3 Further Augmentation

In [12], an encryption on the password verifier was considered and its profile was named as an amplified password file (APF). It might be useful when we consider a hardware security module (HSM) which could decrypt the encrypted verifier in a secure manner [28]. Figure 6 depicts the AMP protocol with APF.

3 Analysis

The current draft of this document may concentrate on the efficiency comparison.

3.1 Security

AMP protocols have been examined and improved in terms of security for a pretty long time. AMP protocols are secure against the password-related

Table 1. Comparison of AMP, AMP2, and SRP5

Avg. Mul. in	AMP		AMP2		SRP5	
$GF(p)$ for:	Client	Server	Client	Server	Client	Server
m	1.5κ	-	1.5κ	-	1.5κ	-
m'	-	-	-	1.5κ	-	-
μ	-	$3\kappa + 1$ $\{\mathsf{sim} : 2\kappa\}$	-	$1.5\kappa + 1$	-	$1.5\kappa + 1$
ν	-	-	-	-	1.5κ	-
ν'	-	-	-	-	$1.5(\ell - \kappa)$	$1.5(\ell - \kappa)$
α or β	1.5κ	$1.5\kappa + 61$	1.5κ	$1.5\kappa + 1$	$1.5\kappa + 22$	$3\kappa + 1$ $\{\mathsf{sim} : 2\kappa\}$
Total	3κ	$4.5\kappa + 62$ $\{\mathsf{sim} : 3.5\kappa + 61\}$	3κ	$4.5\kappa + 2$	$1.5\ell + 3\kappa + 22$	$1.5\ell + 3\kappa + 2$ $\{\mathsf{sim} : 1.5\ell + 2\kappa + 1\}$
Example	$480M$	$782M$ $\{\mathsf{sim} : 621M\}$	$480M$	$722M$	$2038M$	$2018M$ $\{\mathsf{sim} : 1857M\}$

attacks such as on-line guessing, off-line guessing, two-for-one guessing, and server compromise, and to provide perfect forward secrecy. In addition, they were examined with regard to their security against other form of protocol attacks. Formal security analysis [1] will be handled.

3.2 Efficiency

An efficiency analysis can be made in a simple but competitive way by considering the number of expensive operations, for example, the number of multiple precision multiplications in $GF(p)$. We compare AMP and SRP as four-pass protocols, and TP-AMP and PAK-Z as three-pass protocols in the augmented model. Tables 1 and 2 compare those four-pass protocols in general by considering real-time computation and possible pre-computation. Table 3 compares them specifically with a safe prime. Tables 4 and 5 compare those three-pass protocols by considering real-time computation and possible pre-computation. Each table approximates the number of multiplications on average, by assuming the use of a left-to-right binary exponentiation method or a simultaneous exponentiation method [17]. Also a slight computational difference between squaring and multiplication is ignored for convenience.

In Table 1, we can see that AMP and AMP2 run more efficiently than SRP5. Note that SRP5 is based on the work of [25]. AMP and SRP5 can benefit from the simultaneous exponentiation in the server. Also SRP5 can benefit from the left-to-right exponentiation if g is selected as a tiny

Table 2. Comparison of AMP, AMP2, and SRP5 with pre-computation

Avg. Mul. in	AMP		AMP2		SRP5	
$GF(p)$ for:	Client	Server	Client	Server	Client	Server
m	$*$	-	$*$	-	$*$	-
m'	-	-	-	1.5κ	-	-
μ	-	$3\kappa + 1$ $\{\mathsf{sim} : 2\kappa\}$	-	$1.5\kappa + 1$	-	$* + 1$
ν	-	-	-	-	1.5κ	-
ν'	-	-	-	-	$1.5(\ell - \kappa)$	$1.5(\ell - \kappa)$
α or β	1.5κ	$1.5\kappa + 61$	1.5κ	$1.5\kappa + 1$	$1.5\kappa + 22$	$3\kappa + 1$ $\{\mathsf{sim} : 2\kappa\}$
Total	1.5κ	$4.5\kappa + 62$ $\{\mathsf{sim} : 3.5\kappa + 61\}$	1.5κ	$4.5\kappa + 2$	$1.5\ell + 1.5\kappa + 22$	$1.5\ell + 1.5\kappa + 2$ $\{\mathsf{sim} : 1.5\ell + 0.5\kappa + 1\}$
Example	$240M$	$782M$ $\{\mathsf{sim} : 621M\}$	$240M$	$722M$	$1798M$	$1778M$ $\{\mathsf{sim} : 1617M\}$

Table 3. Comparison of AMP, AMP3, and SRP6 - safe prime p

Avg. Mul. in $GF(p)$	AMP		AMP3		SRP6	
for operations	Client	Server	Client	Server	Client	Server
m	1.5ℓ	-	1.5ℓ	-	1.5ℓ	-
μ	-	$1.5\ell + 1.5\kappa + 1$	-	$1.5\ell + 1$	-	$1.5\ell + 1$
ν	-	-	-	-	1.5κ	-
w	21	-	-	-	1	-
α or β	1.5ℓ	$1.5\ell + 61$	1.5ℓ	$1.5\ell + 1$	$1.5\ell + 1$	$1.5\ell + 1.5\kappa + 1$
Total	$3\ell + 21$	$3\ell + 1.5\kappa + 62$	3ℓ	$3\ell + 2$	$3\ell + 1.5\kappa + 2$	$3\ell + 1.5\kappa + 2$
Example	$3093M$	$3374M$	$3072M$	$3074M$	$3314M$	$3314M$

generator, but it should be noted that $g = a^{\frac{p-1}{q}} \bmod p$ for an arbitrary generator a of $\mathbb{Z}_p^*$.

In Table 2, pre-computation is considered for AMP, AMP2, and SRP5. We can see that AMP and AMP2 still run more efficiently than SRP5.

In Table 3, we can see that AMP, AMP3, and SRP6 run with similar computational costs. Note that SRP6 requires p chosen as a safe prime, so we compared them with such a specific prime. The reason for comparing AMP3 with other protocols here is that the ACCEPTABLE$_2$() function could run most efficiently for a prime like the safe prime. AMP may need 20 multiplications on average for computing w but can reduce it by running the binary extended gcd algorithm [5, 17]. Due to the property of safe prime, we can expect further efficiency in using the left-to-right method

Table 4. Comparison of TP-AMP and PAK-Z

Avg. Mul. in $GF(p)$	TP-AMP (AMP3 instance)		PAK-Z (with Y instance)	
for operations	Client	Server	Client	Server
γ	$1.5(\ell - \kappa)$	-	$1.5(\ell - \kappa)$	-
m	$1.5\kappa + 1$	-	$1.5\kappa + 1$	-
m'	-	1	-	-
μ	-	$1.5\kappa + 1$	-	1.5κ
ν	-	-	1.5κ	-
α or β	1.5κ	$1.5\kappa + 1$	$1.5\kappa + 1$	$1.5\kappa + 1$
$\mathsf{Sig}_V(\mu)$	-	-	1.5κ	-
$\mathsf{Verify}_W(\mu, s)$	-	-	-	$3\kappa + 1$ $\{\mathsf{sim} : 2\kappa\}$
Total	$1.5\ell + 1.5\kappa + 1$	$3\kappa + 3$	$1.5\ell + 4.5\kappa + 2$	$6\kappa + 2$ $\{\mathsf{sim} : 5\kappa + 1\}$
Example	$1777M$	$483M$	$2258M$	$962M$ $\{\mathsf{sim} : 801\}$

with a small base g such as a^2. However, the total costs increase for exponentiation because the safe prime may need to compute the operations in a large order subgroup [18].

In Table 4 and 5, we can see that TP-AMP runs more efficiently than PAK-Z. Though TP-AMP is not a direct instance of PAK-Z, it can provide better performance on the framework of PAK in the augmented model.

3.3 Many-for-Many Guessing

On standardization, the two-for-one guessing attack was found and considered on the four-pass protocols such as AMP and SRP. It was agreed that such an attack does matter because one party can have more opportunity for on-line guessing attack than it was assumed initially. So those target protocols were respectively refined against the attack.

In this document, we would like to bring up another (possibly known but often ignored) point (this time, on the three-pass protocols) for discussion. Figure 10 depicts the main concept of this simple problem. For example, an adversary poses as a user having id on machine A, and sends an arbitrary message $\langle id, m\rangle$ to the server, based on her guessed password. Then the server may respond with a message $\langle \mu, k_1\rangle$ in the three-pass protocols while only μ in the four-pass protocols.

Here if the adversary forcibly disconnects it without the server detecting her intention, for example, by pulling out the network cable or

Table 5. Comparison of TP-AMP and PAK-Z with pre-computation

Avg. Mul. in $GF(p)$	TP-AMP (AMP3 instance)		PAK-Z (with Y instance)	
for operations	Client	Server	Client	Server
γ	$1.5(\ell-\kappa)$	-	$1.5(\ell-\kappa)$	-
m	$*+1$	-	$*+1$	-
m'	-	1	-	-
μ	-	$1.5\kappa+1$	-	$*$
ν	-	-	1.5κ	-
α or β	1.5κ	$1.5\kappa+1$	$1.5\kappa+1$	$1.5\kappa+1$
$\mathsf{Sig}_V(\mu)$	-	-	1.5κ	-
$\mathsf{Verify}_W(\mu,s)$	-	-	-	$3\kappa+1$ $\{\mathsf{sim} : 2\kappa\}$
Total	$1.5\ell+1$	$3\kappa+3$	$1.5\ell+3\kappa+2$	$4.5\kappa+2$ $\{\mathsf{sim} : 3.5\kappa+1\}$
Example	$1537M$	$483M$	$2018M$	$722M$ $\{\mathsf{sim} : 561\}$

manipulating the transport layer, then the server should time out this session or count up the failed attempts in the three-pass protocols. Here note that the forced disconnection is different even from an abnormal termination from the perspective of protocol run. However, if the server regards it as a time-out event, the adversary is able to verify one guess per time-out and repeats this for many guesses. Let us call this attack *many-for-many guessing* because the adversary could verify her many guesses for many time-out events, for example, by automatic attempts in software. Figure 10-(a) depicts the case.

If the server regards it as a failed event, it may increase the corresponding failure count by one. So, the adversary is able to lock up the target user's account more easily by sending one message for each count and repeating this up to the limit, than finishing the protocol run. The denial of service could come up in a slightly easier way. Even worse, the adversary can still mount the many-for-many guessing attack by sending many different initiating messages based on different password guesses simultaneously to the server before the time is ended up and the server counts up to the limit, if many server instances could respond simultaneously to the same user. The adversary is able to gather many triples, $\langle m, \mu, k_1 \rangle$, and mount the guessing attacks off-line. Figure 10-(b) depicts the case.

However, in the four-pass protocols, the server is able to regard it as simply a time-out event because the value μ is only given to the adversary who cannot verify her guess from this. For the opposite case to the four-pass protocols (say, the adversary poses as the server), the fresh session must be initiated by the user re-entering *id* and password manually. So, the client can regard it as a failed event to count up, and thus such an attack is negligible.

As a result, if it has not been handled yet, we would like to recommend the cases should be noted in the standard document, with cautions such that the server must regard the time-out event as the failed-event and must not be instantiated for the simultaneous requests of the same *id* in the three-pass protocols.

4 Conclusion

We have summarized the AMP protocols and compared them with closely related protocols such as SRP and PAK-Z in the augmented model.

References

1. M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attack," In *Eurocrypt 2000*, LNCS 1807, pp.139-155, 2000
2. S. Bellovin and M. Merritt, "Encrypted key exchange : password-based protocols secure against dictionary attacks," In *IEEE Symposium on Research in Security and Privacy*, pp. 72-84, 1992
3. S. Bellovin and M. Merritt, "Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password-file compromise," In *ACM Conference on Computer and Communications Security*, pp. 244-250, 1993
4. V. Boyko, P. MacKenzie and S. Patel, "Provably secure password authenticated key exchange using Diffie-Hellman," In *Eurocrypt 2000*, LNCS 1807, pp.156-171, 2000
5. H. Cohen, A. Miyajim, and T. Ono "Efficient elliptic curve exponentiation using mixed coordinate," In *Asiacrypt'98*, LNCS 1514, pp.51-65, 1998
6. W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol.22, no.6, pp.644-654, November 1976
7. O. Goldreich and Y. Lindell, "Session-Key Generation Using Human Passwords Only," In *Cypto 2001*, LNCS 2139, pp.408-432, 2001
8. IEEE P1363-2, "Standard specifications for password-based public key cryptographic techniques," available from `http://grouper.ieee.org/groups/1363/`, December 2002
9. D. Jablon, "Strong password-only authenticated key exchange," *ACM Computer Communications Review*, vol.26, no.5, pp.5-26, 1996
10. J. Katz, R. Ostrovsky, and M. Yung, "Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords ," In *Eurocrypt 2001*, LNCS 2045, pp.475-494, 2001
11. T. Kwon and J. Song, "Secure agreement scheme for g^{xy} via password authentication," *Electronics Letters*, vol.35, no.11, pp.892-893, 27th May 1999

12. T. Kwon, "Authentication and key agreement via memorable password," In *ISOC Network and Distributed System Security Symposium*, February 2001
13. C. Lim and P. Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup," In *CRYPTO 97*, pp.249-263, 1997
14. M. Lomas, L. Gong, J. Saltzer, and R. Needham, "Reducing risks from poorly chosen keys," In *ACM Symposium on Operating System Principles*, pp.14-18, 1989
15. P. MacKenzie, "More efficient password-authenticated key exchange," In *RSA Conference*, Cryptographers Track, LNCS 2020, pp.361-377, 2001
16. P. MacKenzie, "The PAK suite: Protocols for Password-Authenticated Key Exchange," Submission to IEEE P1363.2, April 2002
17. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of applied cryptography*, CRC Press,Inc., pp.517-518, 1997
18. P. van Oorschot and M. Wiener, "On Diffie-Hellman key agreement with short exponents," In *Eurocrypt 96*, pp. 332-343, 1996
19. S. Patel, "Number theoretic attacks on secure password schemes," In *IEEE Symposium on Security and Privacy*, 1997
20. R. Perlman and C. Kaufman, "PDM: A new strong password-based protocol," In *USENIX Security Symposium*, pp.313-321, 2001
21. Phoenix Technologies, Inc., "Research Papers on Strong Password Authentication," available from `http://www.integritysciences.com/links.html`, 2002
22. M. Roe, B. Christianson and D. Wheeler, "Secure sessions from weak secrets," Technical report from University of Cambridge and University of Hertfordshire, 1998, available from `http://www.ccsr.cam.ac.uk/techreports/tr4/index.html`
23. C. Schnorr, "Efficient identification and signatures for smart cards," In *CRYPTO 89*, pp.239-251, 1989
24. M. Scott, *Personal communication*, July 2001
25. Y. Wang, "EC-SRP," Submission to IEEE P1363, June 2001
26. T. Wu, "Secure remote password protocol," In *ISOC Network and Distributed System Security Symposium*, 1998
27. T. Wu, "SRP6: Improvements and refinements to the secure remote password protocol," Unpublished document, October 2002.
28. Ncipher.com, `http://www.ncipher.com`

$$\begin{array}{lcl}
\mathsf{user}[id,\pi] \text{ on } A & & \mathsf{server}[id,\nu = g^u] \text{ on } B \\
 & & \\
x \leftarrow_R \mathbb{Z}_q & & \\
m \leftarrow g^x \bmod p & & \\
 & \xrightarrow{id, m} & \\
u \leftarrow h_0(id,\pi) & & y \leftarrow_R \mathbb{Z}_q \\
e_1 \leftarrow h_1(id,A,B,m) & & e_1 \leftarrow h_1(id,A,B,m) \\
u_1 \leftarrow (xe_1 + u)^{-1} \bmod q & & \mu \leftarrow (m^{e_1}\nu)^y \bmod p \\
 & \xleftarrow{\mu} & \\
e_2 \leftarrow h_2(id,A,B,m,\mu) & & e_2 \leftarrow h_2(id,A,B,m,\mu) \\
w \leftarrow u_1(x + e_2) \bmod q & & \\
\alpha \leftarrow \mu^w \bmod p & & \beta \leftarrow (mg^{e_2})^y \bmod p \\
k_1 \leftarrow h_3(id,A,B,m,\mu,\alpha) & & k_1' \leftarrow h_3(id,A,B,m,\mu,\beta) \\
 & \xrightarrow{k_1} & \\
 & & \text{abort if } k_1 \neq k_1' \\
k_2' \leftarrow h_4(id,A,B,m,\mu,\alpha) & & k_2 \leftarrow h_4(id,A,B,m,\mu,\beta) \\
 & \xleftarrow{k_2} & \\
\text{abort if } k_2 \neq k_2' & & \\
K \leftarrow h_5(id,A,B,m,\mu,\alpha) & & K' \leftarrow h_5(id,A,B,m,\mu,\beta)
\end{array}$$

Fig. 1. AMP

$$\begin{array}{lcl}
\mathsf{user}[id,\pi] \text{ on } A & & \mathsf{server}[id,\nu = g^u] \text{ on } B \\
 & & \\
x \leftarrow_R \mathbb{Z}_q & & \\
m \leftarrow g^x \bmod p & & \\
 & \xrightarrow{id, m} & \\
u \leftarrow h_0(id,\pi) & & y \leftarrow_R \mathbb{Z}_q \\
e \leftarrow h_1(id,A,B,m) & & e \leftarrow h_1(id,A,B,m) \\
 & & m' \leftarrow m^e \bmod p \\
w \leftarrow (xe + u)^{-1}(x+1) \bmod q & & \mu \leftarrow (m'\nu)^y \bmod p \\
 & \xleftarrow{\mu} & \\
\alpha \leftarrow \mu^w \bmod p & & \beta \leftarrow (m'g)^y \bmod p \\
k_1 \leftarrow h_3(id,A,B,m,\mu,\alpha) & & k_1' \leftarrow h_3(id,A,B,m,\mu,\beta) \\
 & \xrightarrow{k_1} & \\
 & & \text{abort if } k_1 \neq k_1' \\
k_2' \leftarrow h_4(id,A,B,m,\mu,\alpha) & & k_2 \leftarrow h_4(id,A,B,m,\mu,\beta) \\
 & \xleftarrow{k_2} & \\
\text{abort if } k_2 \neq k_2' & & \\
K \leftarrow h_5(id,A,B,m,\mu,\alpha) & & K' \leftarrow h_5(id,A,B,m,\mu,\beta)
\end{array}$$

Fig. 2. AMP2

$\mathsf{user}[id, \pi]$ on A		$\mathsf{server}[id, \nu = g^u]$ on B
$x \leftarrow_R \mathbb{Z}_q$		
$m \leftarrow g^x \bmod p$		
	$\xrightarrow{id, m}$	
$u \leftarrow h_0(id, \pi)$		$y \leftarrow_R \mathbb{Z}_q$
$w \leftarrow (x+u)^{-1}(x+1) \bmod q$		$c \leftarrow m\nu \bmod p$
		$\mu \leftarrow g^y \bmod p$ if $\neg\mathsf{ACCEPTABLE}_2(c)$
		$\mu \leftarrow c^y \bmod p$ otherwise
	$\xleftarrow{\mu}$	
		$d \leftarrow mg \bmod p$
		abort if $\neg\mathsf{ACCEPTABLE}_2(d)$
$\alpha \leftarrow \mu^w \bmod p$		$\beta \leftarrow d^y \bmod p$ otherwise
$k_1 \leftarrow h_3(id, A, B, m, \mu, \alpha)$		$k_1' \leftarrow h_3(id, A, B, m, \mu, \beta)$
	$\xrightarrow{k_1}$	
		abort if $k_1 \neq k_1'$
$k_2' \leftarrow h_4(id, A, B, m, \mu, \alpha)$		$k_2 \leftarrow h_4(id, A, B, m, \mu, \beta)$
	$\xleftarrow{k_2}$	
abort if $k_2 \neq k_2'$		
$K \leftarrow h_5(id, A, B, m, \mu, \alpha)$		$K' \leftarrow h_5(id, A, B, m, \mu, \beta)$

Fig. 3. AMP3

$\mathsf{user}[id, \pi]$ on A		$\mathsf{server}[id, \nu = g^u, \gamma' = \gamma^{-1}]$ on B
$x \leftarrow_R \mathbb{Z}_q$		
$u \leftarrow h_0(id, \pi)$		
$\gamma \leftarrow H_1(id, u)$		
$m \leftarrow g^x \gamma \bmod p$		
	$\xrightarrow{id, m}$	
		$y \leftarrow_R \mathbb{Z}_q$
$w \leftarrow (x + u)^{-1}(x + 1) \bmod q$		$m' \leftarrow m\gamma' \bmod p$
		$c \leftarrow m\nu \bmod p$
		abort if $\neg\mathsf{ACCEPTABLE}_2(c)$
		$\mu \leftarrow c^y \bmod p$ otherwise
		$d \leftarrow mg \bmod p$
		abort if $\neg\mathsf{ACCEPTABLE}_2(d)$
		$\beta \leftarrow d^y \bmod p$
		$k_1 \leftarrow h_3(id, A, B, m, \mu, \beta)$
	$\xleftarrow{\mu, k_1}$	
$\alpha \leftarrow \mu^w \bmod p$		
$k_1' \leftarrow h_3(id, A, B, m, \mu, \alpha)$		
abort if $k_1 \neq k_1'$		
$k_2 \leftarrow h_4(id, A, B, m, \mu, \alpha)$		$k_2' \leftarrow h_4(id, A, B, m, \mu, \beta)$
	$\xrightarrow{k_2}$	
		abort if $k_2 \neq k_2'$
$K \leftarrow h_5(id, A, B, m, \mu, \alpha)$		$K' \leftarrow h_5(id, A, B, m, \mu, \beta)$

Fig. 4. TP-AMP (AMP3 instance)

$$\begin{array}{lcl}
\mathsf{user}[id, \pi] \text{ on } A & & \mathsf{server}[id, \nu = g^u, \gamma' = \gamma^{-1}] \text{ on } B \\
 & & \\
x \leftarrow_R \mathbb{Z}_q & & \\
u \leftarrow h_0(id, \pi) & & \\
\gamma \leftarrow H_1(id, u) & & \\
m \leftarrow g^x \gamma \bmod p & & \\
 & \xrightarrow{id, m} & \\
 & & \text{abort if } \neg\mathsf{ACCEPTABLE}_1(m) \\
 & & y \leftarrow_R \mathbb{Z}_q \\
 & & \mu \leftarrow \nu^y \bmod p \\
e \leftarrow h_2(id, A, B, m) & & e \leftarrow h_2(id, A, B, m) \\
w \leftarrow u^{-1}(x + e) \bmod q & & \beta \leftarrow (m\gamma' g^e)^y \bmod p \\
 & & k_1 \leftarrow h_3(id, A, B, m, \mu, \beta) \\
 & \xleftarrow{\mu, k_1} & \\
\alpha \leftarrow \mu^w \bmod p & & \\
k_1' \leftarrow h_3(id, A, B, m, \mu, \alpha) & & \\
\text{abort if } k_1 \neq k_1' & & \\
k_2 \leftarrow h_4(id, A, B, m, \mu, \alpha) & & k_2' \leftarrow h_4(id, A, B, m, \mu, \beta) \\
 & \xrightarrow{k_2} & \\
 & & \text{abort if } k_2 \neq k_2' \\
K \leftarrow h_5(id, A, B, m, \mu, \alpha) & & K' \leftarrow h_5(id, A, B, m, \mu, \beta)
\end{array}$$

Fig. 5. TP-AMP2

$$\begin{array}{lcl}
\mathsf{user}[id, \pi] \text{ on } A & & \mathsf{server}[id, \nu = g^{\frac{u}{v}}, \tau] \text{ on } B \\
 & & \\
x \leftarrow_R \mathbb{Z}_q & & \\
m \leftarrow g^x \bmod p & & \\
 & \xrightarrow{id, m} & \\
u \leftarrow h_0(id, B, \pi) & & y \leftarrow_R \mathbb{Z}_q \\
e_1 \leftarrow h_1(id, A, B, m) & & e_1 \leftarrow h_1(id, A, B, m) \\
 & & v \leftarrow \tau + s \bmod q \\
u_1 \leftarrow (xe_1 + u)^{-1} \bmod q & & \mu \leftarrow (m^e \nu^v)^y \bmod p \\
 & \xleftarrow{\mu} & \\
e_2 \leftarrow h_2(id, A, B, m, \mu) & & e_2 \leftarrow h_2(id, A, B, m, \mu) \\
w \leftarrow u_1(x + e_2) \bmod q & & \\
\alpha \leftarrow \mu^w \bmod p & & \beta \leftarrow (mg^{e_2})^y \bmod p \\
k_1 \leftarrow h_3(id, A, B, m, \mu, \alpha) & & k_1' \leftarrow h_3(id, A, B, m, \mu, \beta) \\
 & \xrightarrow{k_1} & \\
 & & \text{abort if } k_1 \neq k_1' \\
k_2' \leftarrow h_4(id, A, B, m, \mu, \alpha) & & k_2 \leftarrow h_4(id, A, B, m, \mu, \beta) \\
 & \xleftarrow{k_2} & \\
\text{abort if } k_2 \neq k_2' & & \\
K \leftarrow h_5(id, A, B, m, \mu, \alpha) & & K' \leftarrow h_5(id, A, B, m, \mu, \beta)
\end{array}$$

Fig. 6. AMP with Amplified Password File

$\mathsf{user}[id, \pi]$ on A		$\mathsf{server}[id, \nu = g^u]$ on B
$x \leftarrow_R \mathbb{Z}_q$		
$m \leftarrow g^x \bmod p$		
	$\xrightarrow{id, m}$	
$u \leftarrow h_0(id, \pi)$		$y \leftarrow_R \mathbb{Z}_q$
$\nu \leftarrow g^u \bmod p$		$\nu' \leftarrow (h_1(\nu))^r \bmod p$
$\nu' \leftarrow (h_1(\nu))^r \bmod p$		$\mu \leftarrow \nu' g^y \bmod p$
	$\xleftarrow{\mu}$	
$w \leftarrow h_2(\mu)$		$w \leftarrow h_2(\mu)$
$w_1 \leftarrow x + uw \bmod q$		
$\alpha \leftarrow (\mu \nu'^{-1})^{w_1} \bmod p$		$\beta \leftarrow (m\nu^w)^y \bmod p$
$k_1 \leftarrow h_3(id, B, \alpha)$		$k_1' \leftarrow h_3(id, B, \beta)$
	$\xrightarrow{k_1}$	
		abort if $k_1 \neq k_1'$
$k_2' \leftarrow h_4(id, B, \alpha)$		$k_2 \leftarrow h_4(id, B, \beta)$
	$\xleftarrow{k_2}$	
abort if $k_2 \neq k_2'$		
$K \leftarrow h_5(id, B, \alpha)$		$K' \leftarrow h_5(id, B, \beta)$

Fig. 7. SRP5

$\mathsf{user}[id, \pi]$ on A		$\mathsf{server}[id, \nu = g^u]$ on B
$x \leftarrow_R \mathbb{Z}_q$		
$m \leftarrow g^x \bmod p$		
	$\xrightarrow{id, m}$	
$u \leftarrow h_0(id, \pi)$		$y \leftarrow_R \mathbb{Z}_q$
$\nu \leftarrow g^u \bmod p$		$\mu \leftarrow 3\nu + g^y \bmod p$
	$\xleftarrow{\mu}$	
$w \leftarrow h_1(m, \mu)$		$w \leftarrow h_1(m, \mu)$
$w_1 \leftarrow x + uw \bmod q$		
$\alpha \leftarrow (\mu - 3\nu)^{w_1} \bmod p$		$\beta \leftarrow (m\nu^w)^y \bmod p$
$k_1 \leftarrow h_3(id, B, \alpha)$		$k_1' \leftarrow h_3(id, B, \beta)$
	$\xrightarrow{k_1}$	
		abort if $k_1 \neq k_1'$
$k_2' \leftarrow h_4(id, B, \alpha)$		$k_2 \leftarrow h_4(id, B, \beta)$
	$\xleftarrow{k_2}$	
abort if $k_2 \neq k_2'$		
$K \leftarrow h_5(id, B, \alpha)$		$K' \leftarrow h_5(id, B, \beta)$

Fig. 8. SRP6

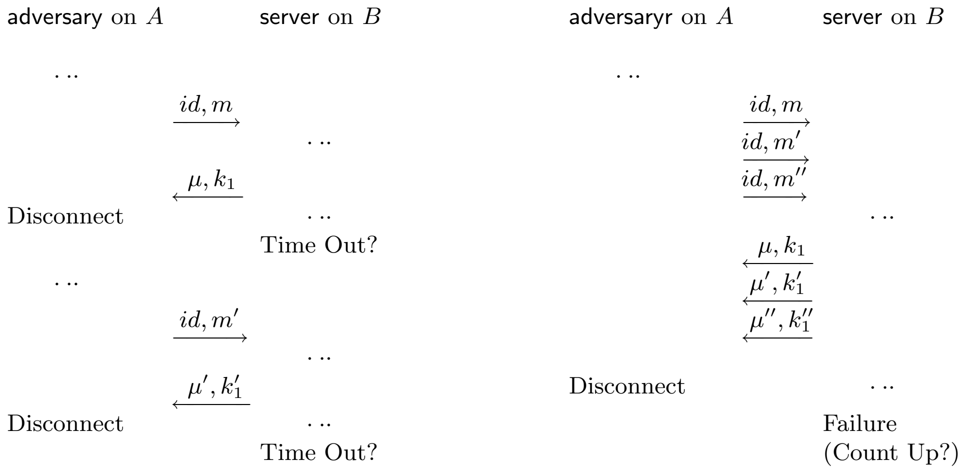
$\mathsf{user}[id, \pi]$ on A		$\mathsf{server}[id, \gamma' = \gamma^{-1}, W, V' = H_2(id, \pi) \oplus V]$ on B
$x \leftarrow_R \mathbb{Z}_q$		
$\gamma \leftarrow H_1(id, u)$		
$m \leftarrow g^x \gamma \bmod p$		
	$\xrightarrow{id, m}$	
		abort if $\neg\mathsf{ACCEPTABLE}_1(m)$
		$y \leftarrow_R \mathbb{Z}_q$
		$\mu \leftarrow g^y \bmod p$
		$\beta \leftarrow (m\gamma')^y \bmod p$
		$a' \leftarrow H_6(id, B, m, \mu, \beta, \gamma')$
		$a \leftarrow a' \oplus V'$
		$s'' \leftarrow H_7(id, B, m, \mu, \beta, \gamma')$
		$k_1 \leftarrow h_3(id, B, m, \mu, \beta, \gamma')$
	$\xleftarrow{\mu, k_1, a}$	
$\alpha \leftarrow \mu^x \bmod p$		
$\gamma' \leftarrow \gamma^{-1} \bmod p$		
$k_1' \leftarrow h_3(id, B, m, \mu, \alpha, \gamma')$		
abort if $k_1 \neq k_1'$		
$a' \leftarrow H_6(id, B, m, \mu, \alpha, \gamma')$		
$V' \leftarrow a' \oplus a$		
$V \leftarrow H_2(id, \pi) \oplus V'$		
abort if $\neg\mathsf{VALID}(V)$		
$s \leftarrow \mathsf{Sig}_V(\mu)$		
$s'' \leftarrow H_7(id, B, m, \mu, \alpha, \gamma')$		
$s' \leftarrow s'' \oplus s$		
	$\xrightarrow{s'}$	
		$s \leftarrow s'' \oplus s'$
		abort if $\mathsf{Verify}_W(\mu, s) = 0$
$K \leftarrow h_5(id, B, m, \mu, \alpha, \gamma')$		$K' \leftarrow h_5(id, B, m, \mu, \beta, \gamma')$

$\rhd$ $V = \langle v, \mathsf{mac}(v) \rangle$; $W = g^v \bmod p$ where $v \leftarrow_R \mathbb{Z}_q$

$\rhd$ $\mathsf{Sig}_V(\mu)$ $\{c \leftarrow_R \mathbb{Z}_q;\ \delta \leftarrow g^c \bmod p;\ e \leftarrow H_0'(id, B, m, \mu, \alpha, \gamma', \delta);$
$\sigma \leftarrow c - ev \bmod q;\ s \leftarrow \langle e, \sigma \rangle\}$

$\rhd$ $\mathsf{Verify}_W(\mu, s)$ $\{\zeta \leftarrow g^\sigma W^e \bmod p;\ \eta \leftarrow H_0'(id, B, m, \mu, \beta, \gamma', \zeta);$
return 0 if $e \neq \eta\}$

Fig. 9. PAK-Z ($\rhd$ specifies Y instance)



(a) Serial Attempts

(b) Parallel Attempts

Fig. 10. Many-for-Many Guessing