WITH VAULTED VOICE VERIFICATION MY VOICE IS MY KEY

R.C. Johnson *and* *Terrance E. Boult*

Univ. of Colorado Colorado Springs and Securics, Inc.
Colorado Springs, CO, USA

ABSTRACT

Those who handle sensitive information from time to time need a device that can communicate securely. They also need the ability to verify the recipient of the information. For such secure communication to take place, they must securely exchange a key, often with someone they do not already know.

Biometrics have been gaining widespread adoption in an effort to verify the end users identity. We extend this to key exchange. Vaulted Voice Verification, a recently introduced voice-based biometric protocol, has been shown to securely and remotely verify a user while also maintaining the privacy of the user. However Vaulted Voice Verification as originally proposed was not well suited for the exchange of larger keys. We present an index-based Vaulted Voice Verification which significantly reduces communication overhead and allows the transmission of keys that are suitable for biometrically authenticated secure communication.

Index Terms— Vaulted Voice Verification, biometrics, security, privacy, speaker verification

1. INTRODUCTION AND RELATED WORK

Smart phones have gained widespread adoption, from friends surfing while calling each other socially, to first responders in the field communicating with headquarters. Integrated multi-purpose devices are replacing dedicated phone, camera, web and gaming devices. While most of the communication that occurs via smart phones is benign, some of it is sensitive and needs to be securely transmitted. For those who handle sensitive information, there also exists a need for a device that can communicate securely. For such people, using their existing smart phone to make secure communication calls can be very appealing.

For a mobile device to conduct secure communications, that device must first have a secure key. If two people are to communicate, how can they both securely exchange a key and be sure of the identity of the person with whom they are communicating? The answer: by incorporating biometrics into the process of key exchange. This leads to the question: how can biometrics be incorporated in a way that maintains security and trust while preserving privacy?

Over the years, much research has gone into the use of biometrics in generating or storing keys, [1, 2, 3, 4, 5, 6, 7, 8], with a few works on biometric verified key exchange [2, 3, 6]. One of the biggest hurdles found with using biometrics for secure key generation is that biometrics are inherently noisy. In general, these and other works attempt to create a biometric template of one form or another to solve this problem.

Some work of note includes fuzzy extractors and fuzzy commitment schemes [5, 9, 10, 11]. The general idea behind these schemes is to hide real data among lots of chaff data during enrollment; "locking" the data in a "vault." During verification in these types of schemes, if the biometrics of the verifying user can match enough of the real data points, the key is "released" from the "vault." While such schemes seemingly solve the problem of creating exact keys from noisy data, they are susceptible to attacks [12] and they cannot solve the problem of remote key exchange.

Other work involves generating protected templates, which can be seen as a type of key [13, 14, 1]. While such schemes can seemingly solve the problem of using biometrics to generate a key, they do not solve the problem of remote key exchange that also preserves privacy. Also, because they can not do any sort of challenge response, the two parties exchanging keys can not explicitly trust each other without violating privacy. In these systems, either the keys are generated or released. However, the keys they manage are too small to be useful for RSA public keys.

To solve the problem of using biometric security to remotely verify identification and also preserve privacy, we look to Vaulted Voice Verification. However, Vaulted Voice Verification, as originally proposed, was not well suited for the exchange of keys large enough for secure communication. This is because, similar to other biometric key exchange schemes, each interaction with the user generated only a small number of bits. Hundreds of bits are needed to generate keys that are secure enough for practical use. The originally proposed Vaulted Voice Verification would require far too many user interactions to be practical. To overcome this, this paper shows how to create an index-based Vaulted Voice Verification capable of generating keys large enough for RSA public keys.

Vaulted Voice Verification, introduced in [15], is a voice-based extension of a technique called Vaulted Verification, which was developed for privacy and security enhanced biometric identification [2, 16]. Vaulted Voice Verification is an extension of the Vaulted Verification concept, combining it with with technology for speaker identification and with recognition techniques from the voice community [17, 18, 19, 20]. The result is a secure protocol that is able to remotely identify a user via their voice while also maintaining their privacy.

As with other protocols, Vaulted Voice Verification utilizes both an enrollment and a verification stage. In general, the enrollment and verification stages of Vaulted Voice Verification are easy to communicate. During the enrollment stage, the user is presented with questions to answer or phrases to repeat. The user submits answers/responses in the form of voice samples which are then turned into models and stored for later comparison. During the verification stage, the user is asked a subset of the same questions and their responses must correspond to the correct answer as stored during enrollment. The remainder of this section will include a high level overview of the originally proposed Vaulted Voice Verification protocol. For a more in-depth explanation, refer to [15].

1.1. Enrollment

First, the user enters their information into their device, which then generates keys for the user. The device then prompts the user with questions to answer or words to repeat. The user responds to each prompt as requested. For each of these prompts, a real model and chaff models are created so that the chaff models are similar to the real. The device then encrypts the models using the public key of the user. The public key of the user is used here so that only the user can decrypt the models with their private key.

Then the models are sent to the server for hashing. The server stores the hashes and discards the model information. Because of this, the encrypted models can remain on the client device until a verification is requested.

The server then sends a response notification to the client device to let it know that the enrollment is complete. The client device notifies the user of a successful enrollment.

1.2. Verification

The user first inputs their information into their device, and the device uses that information to send a verification request to the server. The server decrypts the template, verifies the hash and generates an i.i.d. binary string. The server then selects a subset of the models and shuffles them according to the string. Then the server sends the shuffled models to the client.

After receiving the shuffled models, the client device decrypts them. The client device then prompts the user with the questions or words to repeat for each model group. The user responds to each prompt, generating a new model each time. The client device then makes a comparison between the models sent from the server and the newly generated models. This comparison allows the client device to guess real models from the chaff ones and unscramble the challenge. As the client device unscrambles the challenge, it generates a response string. The completed response string is sent to the server for verification.

Upon receiving the response string from the client device, the server compares it against the string it generated for the challenge. If the strings match within some tolerance, the server responds with an accept. If, however, the strings do not, the server rejects the verification attempt.

2. INDEX-BASED SECURE KEY EXCHANGE

The originally proposed Vaulted Voice Verification protocol is able to securely and remotely verify an individual using voice as a biometric identifier. In this work, we look to use the basic ideas from Vaulted Voice Verification to create a protocol that is able to securely and remotely identify an individual while exchanging keys that are suitable for secure communication, all while radically reducing communication overhead.

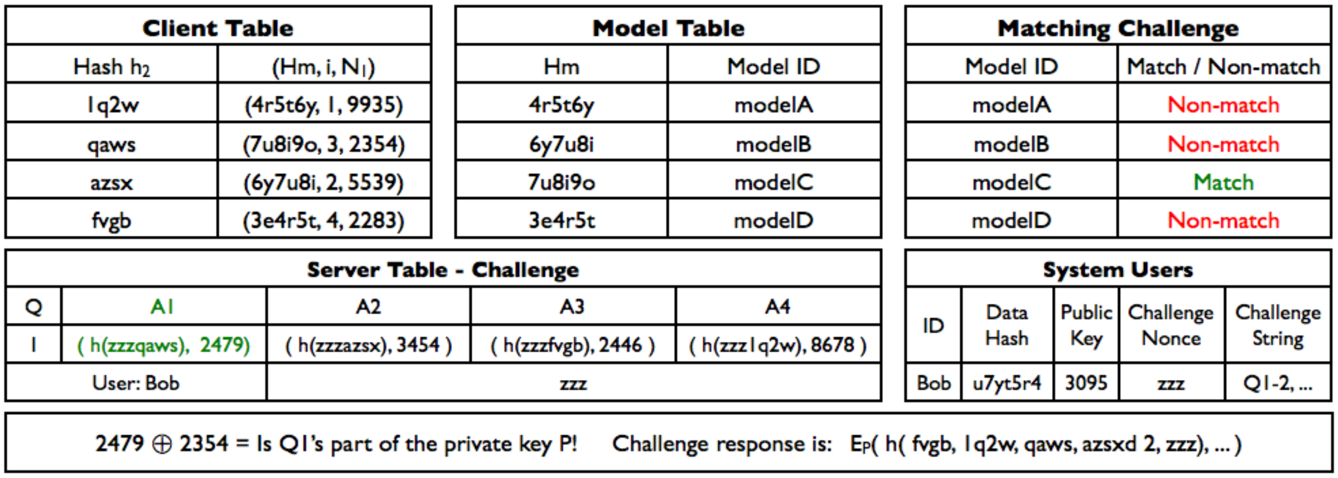
As with Vaulted Voice Verification, our index-based protocol for secure key exchange has both an enrollment and a verification stage. What makes this protocol truly novel is its use of indexed tables instead of sending models between the server and the client. We will examine the construction of the five different types of tables that must be created before we detail how the protocol is used for secure key exchange. The tables are as follows: model table, client table, server table, server challenge table, matching table and user table. During matching, the biometrics are combined with these tables to generate a set of results for the matching challenge. The five tables with sample data are shown in Fig. 1.

2.1. Index-based Enrollment

The enrollment process for our new index-based version of Vaulted Voice Verification is similar to the enrollment of the originally proposed Vaulted Voice Verification. As shown in Fig. 2, the enrollment process occurs in steps. In step 1, the user enters their ID and password. One option is for the user to supply both of them; another option is to base their generation on the client device (i.e. phone number and hardware ID), making the enrollment device specific.

In step 2, the client device generates the RSA key pair, $K_{\text{priv}}, K_{\text{pub}}$ for the user. In step 3, the client device selects questions from a larger list, asks the user the questions, and the user responds. In step 4 the client generates models based on the user's responses, one model per response.

Once the models are generated, the process moves to step 5, in which the client device generates the model table described in Fig. 1. It takes the private key K_{priv} from step 2 and a set of generated random pad R and generates a xored version of the key $\hat{K} = K_{\text{priv}} \oplus R$. It breaks $\hat{K}$ into smaller segment, e.g. 32 bit segments, and stores the components in the client table as N_i associated with hash value $h_{2,i}$ associated with correct model i and associating a random N_i with chaff pairs. It encrypts the model and client table with the user-device password. It takes the associated segment of the pad R_i, and adds the pair $(h_{2,i}, R_i)$ into a temporary server table. It also stores the question string for each pair – note the client table does NOT contain the question strings. Then the temporary server table is encrypted using the server's public key before sending it to the server. The server then receives the table, decrypts it, and generates a hash based on the content. In step 6, the server stores the generated hash as the data hash for



Client Table	
Hash h_2	(Hm, i, N_1)
1q2w	(4r5t6y, 1, 9935)
qaws	(7u8i9o, 3, 2354)
azsx	(6y7u8i, 2, 5539)
fvgb	(3e4r5t, 4, 2283)

Model Table	
Hm	Model ID
4r5t6y	modelA
6y7u8i	modelB
7u8i9o	modelC
3e4r5t	modelD

Matching Challenge	
Model ID	Match / Non-match
modelA	Non-match
modelB	Non-match
modelC	Match
modelD	Non-match

Server Table - Challenge				
Q	A1	A2	A3	A4
1	(h(zzzqaws), 2479)	(h(zzzazsx), 3454)	(h(zzzfvgb), 2446)	(h(zzz1q2w), 8678)
User: Bob		zzz		

System Users				
ID	Data Hash	Public Key	Challenge Nonce	Challenge String
Bob	u7yt5r4	3095	zzz	Q1-2, ...

2479 ⊕ 2354 = Is Q1's part of the private key P!   Challenge response is: E_P(h(fvgb, 1q2w, qaws, azsxd 2, zzz), ...)

Fig. 1. Index-based Vaulted Voice Verification Tables. The Model Table (upper-middle) consists of model files and their associated hashes; it contains no information linking models to challenges/questions. The Client Table (upper-left) consists of a triple of (hash of a model, an index value, a nonce N_1), and a hash of the triple (h_2). The Server Table is not shown, but is used to generate the Server Table - Challenge (middle-left) which contains a question, a user ID, a challenge nonce, and a set of doubles, four in this example. The doubles contain a hash created from h_2 hashed with the challenge nonce, and a different nonce N_2. The Matching Challenge (upper-right) is not saved; it is a visual representation indicating the matching and non-matching models for this challenge.The System Users table (middle-right) contains an ID, a data hash of the complete template, a public key, a challenge nonce, and a challenge string. The private key (bottom) results from applying an XOR to N_2 and N_1 across all questions.The proper response to the challenge results from encrypting h_2's in correct order for a question with the number of shifts and the challenge nonce, for each question, with the private key P.

the user in the system users table. Then the server adds an enrollment date/time/ID, encrypts the resultant "server table" with it's public key, sends it to the user in step 7 and may then delete its local copy of the server table. The data is encrypted with the public key of the server so the client is not able to access the content, but gives a copy to the client so the server does not retain any information other than the hash and user ID, reducing any risk from insiders trying to decrypt the data. In steps 8 and 9, the client stores the server table, deletes the temporary server table and notifies the user that the enrollment is complete.

2.2. Index-based Verification

The verification process for our new index-based version of Vaulted Voice Verification is a variant of the one originally proposed. Unlike the original, however, this manages only indices and extracts a private key.

In step 1, the ID and password are obtained: either the user enters them or the device generates them. Next, the client device uses the password to attempt to decrypt the tables and model files. If successful, the ID of the user and the encrypted server table are sent to the server to begin the verification process.

The server receives the request, and, in step 3, decrypts the table and hashes it to compare the request against the previously stored hash of the table that was sent to the given user. Then the server selects a subset and permutation of the questions from the server table. The server then generates a string of random numbers for which each number is less than the total number of available answers for a given question. The length of the string equals the number of selected questions. Then the server cyclically shifts the rows of the selected questions from the server table according to the bit string. It generates a challenge nonce and appends it to the first elements of row pairs, hashing the results. The resulting shifted/rehashed subset of entries plus the challenge nonce form the server challenge table. Once the challenge table is complete, the server sends the table to the client, shown in step 4.

Using the client table, the first value from the double in the challenge table and the nonce from the challenge table, the client searches the challenge table for matching entries, then produces a triple with the hash Hm, index i, and potential key fragment $N_i \oplus R_I$). Using the triples and the model table, the client device locates the associated model files, step 5. In step 6, the client device uses the questions from the challenge table to prompt the user, to which the user responds. The client device generates models for the questions from the answer given by the user. Similar to Vaulted Voice Verification as originally proposed, in step 8, the client device then compares the models against the possible models. This allows for the identification of the correct triples.

Obtaining the correct triples allows for the identification of the shift of the rows of the challenge. The client device accomplishes this by examining the index variable of each triple. If, for example, the triple contains an index of 3, yet it is in position 1 of the matching challenge table, then the

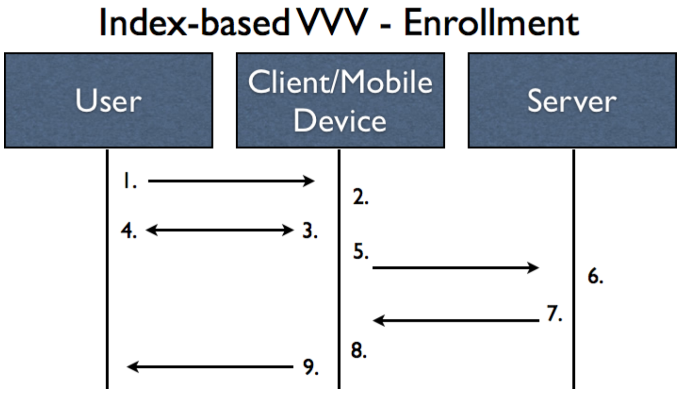


Fig. 2. Index-based Vaulted Voice Verification: Enrollment. 1. Input user ID and password. 2. Generate User's RSA keys. 3. Ask user questions/phrases. 4. Respond to questions/phrases. 5. Generate and encrypt models. Generate Client, Model and Temp. Server Tables. Encrypt Temp. Server Table with Server's public key. Send ID and Temp. Server Table. 6. Receive ID and Server Table. Decrypt, hash, and encrypt Table with private key. Store ID and hashes for new user. 7. Send encrypted Server Table. 8. Delete temporary Server Table. Store encrypted Server Table. 9. Inform user of successful enrollment.

server applied a shift of 2. Client device does this for each row, generating a string of presumed shifts. The client device can also take the potential partial key fragments and combine them to obtain the private key. This could be simple reordering and concatenation, xoring, or, ideally, it can be accomplished using an error correcting polynomial.

From here the client device computes its response. The response from the client device to the server comprises information only available after generating the presumed shifts. The client device responds to each challenge from the server with a message encrypted with the recovered private key K_{priv}. The client device then encrypts a series of hashes (h_3) generated from each question. Each h_3 results from hashing the correctly ordered hashes from the Client Table, the applied shift for the question, and the challenge nonce. The client device then sends this encrypted message to the server for verification.

The server receives the response, as shown in step 9. The server then verifies that the responses are correct by decrypting the message using the associated public key K_{pub} that was stored at enrollment. The server recreates the expected hashes and compares them with the decrypted response. If correct, the server sends a response message encrypted with server's private key then the users public key K_{pub}. The response may include a session encryption key for secure streaming communication, or the two can use the now verified public/private key for exchanges. Thus, a key that depends on the users biometric and password is recreated and exchanged such that the biometric and password never leave the device, providing increased security and no loss in privacy.

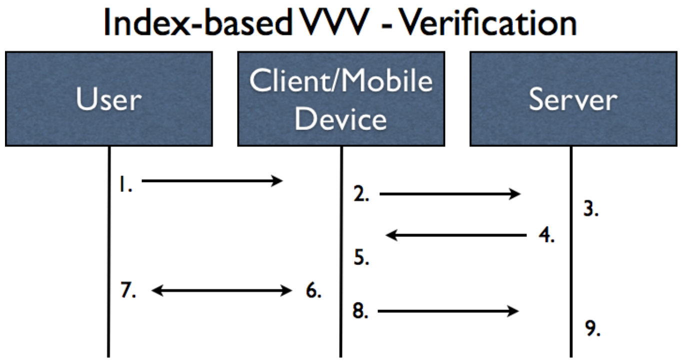


Fig. 3. Index-based Vaulted Voice Verification: Verification. 1. Input user ID and password. 2. Use password to decrypt Saved Tables and Model files. Send ID and encrypted Server Table to request verification. 3. Decrypt and hash Server Table and verify against stored hash. Chose questions from within Server Table. Generate random string S. Shift rows of Server Table according to S, creating Challenge Table. Generate Challenge nonce N and hash entries with nonce. 4. Send Challenge Table to Client. 5. Use Client Table to turn Challenge Table entries into (Model, Index, Nonce). Locate models associated with each question. 6. Ask user questions. 7. Respond to questions. 8. Identify correct models via Responses. Unscramble Challenge using correct models. Compute Responses. Send a message using the generated Response as the Key. 9. Verify response.

3. EXPERIMENTAL ANALYSIS

The focus of this research is the exchange of keys that are biometrically authenticated and a reduction of communication overhead. We require a design where the biometrics are required both to release the key and to properly respond to the remote server/users challenge. Because of this, we discuss results in terms of suitablity of key exchange and the amount of communication overhead required for the construction/exchange of such keys using our new scheme vs. Vaulted Voice Verification as originally proposed.

Both the originally proposed Vaulted Voice Verification and our new index-based protocol depend on generated models and hashes. Both versions of the protocol also require some amount of storage on both the client device and the server. The primary bottleneck for Vaulted Voice Verification as originally proposed is the communication overhead of transferring model files between the server and the client device. The bottleneck, in terms of key generation, is the amount of questions required to generate enough bits for a suitable key and the fact that the server would have enough information to impersonate (via questions) the user. These are all addressed by the novel design described in this paper.

For our tests, we used Gaussian Mixture Models (GMMs). For each model there exists two files: a file containing the means and a file containing the variances. The combined file size for each model, including both files, is 205 kilobytes (K). For generating the hashes we used SHA-512, each taking up 64 bytes. All nonces used are also 512 bit random strings – much bigger than needed but this allowed more consistent coding. In our testing, we consider a challenge-response pair to consist of a question, containing four possible models for the answer, and a response, containing one model for comparison. We assume that the size of the username and questions themselves are negligible and are therefore not mentioned in this discussion. We further assume that the RSA encryption applied during communication has a constant effect on the amount of the data transmitted, and can therefore be left out of the discussion.

In the remainder of this section, we will discuss the difference in the communication overhead between the two versions of the protocol; the difference in required storage space on the the server and client device; and the abilities of each version of the protocol to generate/protect keys and the security.

3.1. Communication Overhead

The term communication overhead encompasses many different things. In this work, we define it as the amount of data transmitted between the client device and the server during an enrollment and a verification of a single challenge-response pair. We examine the difference in communication overhead between Vaulted Voice Verification as originally proposed and our new index-based version of the protocol.

During enrollment in the original protocol, four models are generated and sent to the server, for a total of 820K of data transmission. During verification, the four models are sent twice; once from the client device to the server so the server can scramble their order, and then back to the client device from the server in the form of a challenge. This process creates another 1,640K of data transmission per question. So, for enrollment and verification of one challenge-response pair in Vaulted Voice Verification as originally proposed, it takes roughly 2,460K of data transmission.

During enrollment of our new index-based protocol, a temporary server table containing eight hashes is sent to the server from the client device and an encrypted server table containing eight hashes and a nonce is returned. That is 17 values of 64 bytes each; totaling 1,088 bytes, or roughly 1K. During verification, an encrypted server table is first sent to the server from the client device (eight hashes and a nonce), then a challenge table is sent back (eight hashes and a nonce), and then an encrypted response comprising of four hashes, a single number and a nonce is finally sent to the server. That is 20 hashes, 3 nonces and a single number; totaling 1,476 bytes, roughly 1.5K. So, for enrollment and verification of one challenge-response pair for our new index-based protocol, it takes roughly 2,564 bytes, 2.5K.

Our new index-based protocol transmits 0.1% of the data that the original Vaulted Voice Verification does, i.e. it reduces communication overhead by 1000x. From this, it can be seen that the communication overhead in Vaulted Voice Verification as originally proposed is orders of magnitude larger than that of our new index-based protocol.

3.2. Storage Requirements

Similar to communication overhead, the storage requirements of both protocols must be defined. In this work, we define the storage requirements by the size of the data being stored. We only pay attention to final storage and not temporary or intermediate storage requirements.

When enrollment is complete in the original protocol, the client device contains four models and the server contains a hash of the template. This generates 820K of required storage on the client device and 64 bytes on the server. During verification, no extra data is stored. The total rounded storage requirement for the originally proposed Vaulted Voice Verification protocol is 821K.

When enrollment is complete in our new index-based version of the protocol, there exists a model table, a client table and an encrypted server table on the client device, and a user table on the server. The model table contains 4 models and 4 hashes, 820K + 256 bytes, for a rounded total of 821K. The client table contains 4 hashes and 4 triples, 256 bytes + $4 * (64 + 4 + 64)$, for a rounded total of 784 bytes; 0.7K. The encrypted server table contains 8 hashes and a nonce, 512 + 64, for a rounded total of 576 bytes; 0.5K This gives an enrollment storage for the client device with our index-based version of 822K. The server contains a table for the user. This user table contains a hash of the data, 64 bytes, a public key, 64 bytes, a challenge nonce, 64 bytes, and a challenge string, 1 number, for a rounded total of 196 bytes. This gives an overall enrollment storage for our index-based version of 823K. No extra data is generated for storage at the completion of verification for our index-based version of the protocol. Thus, the number remains 823K.

As shown here, the difference in necessary storage requirements is negligible between the two versions of the protocol. While our new index-based version does contain additional tables compared to the original version, the size of the tables is negligible.

3.3. Key Management and challenge strength

The last area of focus in our experiments is key management and challenge strength. For this, we look at the tradeoff between system usability and the ability to generate/manage keys and challenges of a given size for both protocols. That is, we examine how many user interactions are required to generate a key, release a key, and how much security does that key or challenge actually provide. We define a user interaction as the user being asked and then responding to a single question. We consider each question and its associated responses to be independent. We consider generating keys suitable for RSA. Because we use SHA-512 in combination with RSA to generate the keys [21], 512 is the minimum number of bits required and the minimum size RSA keys generated is 1,024. For this discussions, we define usability as 20 or fewer interactions during enrollment/verification.

With Vaulted Voice Verification as originally proposed, every user interaction produced two bits, considering four possible choices. With this, to generate 512 bits would require 256 user interactions. This is clearly not acceptable. Even if the number of choices per question increased to 8, the resulting 64 interactions required renders the system nearly useless. While increasing the amount of choices per question seems a plausible solution, it is bounded by the discriminability of the models and therefore can not be relied upon as a valid solution.

4. CONCLUSION

With our new index-based Vaulted Voice Verification, each interaction generates far more bits than in the original Vaulted Voice Verificationprotocol, using the added fields in the index tables. This results occurs because responses are created from hashes and indexes rather than binary (or 4-choice) decisions. Our index-based protocol generates an RSA private key for the final response, with a wide range of usbility/security choices. One could, in theory, have a 512 bit key even with just a single challenge question. Remember, the key is already protected by the users password; our challenge is about added security *beyond the standard password protection of the private key*. In addition, the challenge is about how difficult is it for someone with knowledge of the password and the private key to still impersonate the user or the owner to "share" the key. Having multiple questions allows us to split the private key into set of scrambled components K_{priv} which must be properly ordered and combined with the appropriate data from the challenge R_i. R_i is not stored locally, so, even when it comes in from the server, the association is unknown. Recovering the key and returning the correct response, given all passwords and the server's encryption keys, is still $O(2^{(2q)})$, where q is the number of challenge questions. If we assume there are also k bits of user password security, and s bit of server key security, then the new index system protection is $k + s + 2q$ overall, of which $2q$ bits of "biometric identity" security.

As shown, our new index-based version of the Vaulted Voice Verification protocol is able to manage keys suitable for biometrically authenticated secure communication and public-key operations. This results from decoupling the models from the stored question and replacing that link with a novel, table-based index scheme.

Our index-based version of the Vaulted Voice Verification protocol supports keys of sufficient size for use in RSA keys, while allowing variable levels of biometric identity security in the challenge. This new form allows standard PKI/public-key operations to be tied to biometric verification. Importantly, the novel design also overcomes the problem in the original Vaulted Voice Verification protocol wherein the sever challenge was the only "key," hence someone with sufficient access to the server might impersonate the user. In our new design, the use of private-key signed return values means, even with full access to the server and every communication, one cannot impersonate the user. Thus, when a key is shared by this approach, both parties have strong assurance the correct user is in possession of the remote device and is not being impersonated.

5. REFERENCES

[1] K. Inthavisas and D. Lopresti, "Secure speech biometric templates for user authentication," *The institution of Engineering and Technology Biometrics*, February 2012.

[2] Terrance E. Boult Michael J. Wilber, Walter J. Scheirer, "Privv: Privae remote iris-authentication with vaulted verification," *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[3] W Scheirer, Bill Bishop, and T Boult, "Beyond pki: The biocryptographic key infrastructure," in *IEEE Int. Wksp Information Forensics and Security (WIFS)*. IEEE, 2010, pp. 1–6.

[4] A.T.B. Jin, D.N.C. Ling, and A. Goh, "Biohashing: two factor authentication featuring fingerprint data and tokenised random number," *Pat. Rec.*, vol. 37, no. 11, pp. 2245–2255, 2004.

[5] A. Juels and M. Sudan, "A fuzzy vault scheme," *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 237–257, 2006.

[6] WJ Scheirer and TE Boult, "Bio-cryptographic protocols with bipartite biotokens," in *Biometrics Symposium, 2008. BSYM'08*. IEEE, 2008, pp. 9–16.

[7] N.K. Ratha, S. Chikkerur, J.H. Connell, and R.M. Bolle, "Generating cancelable fingerprint templates," *IEEE Trans. PAMI*, vol. 29, no. 4, pp. 561–572, 2007.

[8] Lee-Ying Chong Andrew Beng Jin Teoh, "Secure speech template protection in speaker verification system," *Speech Communication*, vol. 52, no. 150-163, 2010.

[9] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors and cryptography, or how to use your fingerprints," in *Proc. Eurocrypt*, 2004, vol. 4.

[10] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.

[11] X. Boyen, "Reusable cryptographic fuzzy extractors," in *Proc. 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 82–91.

[12] W. Scherier and T. Boult, "Cracking fuzzy vaults and biometric encryption," *Proc. of the 2007 Biometrics Symposium, at Biometrics Consortium Conf. (BCC)*, 2007.

[13] Eliza Yingzi Du, Kai Yang, and Zhi Zhou, "Key incorporation scheme for cancelable biometrics," in *J. of Information Security*, October 2011, pp. 185–194.

[14] K Saraswathi, Dr R Balasubramanian, and B Jayaram, "Iris biometrics based authentication and key exchange system," *IACSIT Int. J. of Engineering and Technology*, vol. 3, no. 1, 2011.

[15] R.C. Johnson, Walter J. Scheirer, and Terrance E. Boult, "Secure voice based authentication for mobile devices: Vaulted voice verification," *Proc. SPIE, Biometric and Surveillance Tech. for Human and Activity Identification*, vol. 8712, 2013.

[16] Michael J. Wilber and Terrance E. Boult, "Secure remote matching with privacy: Scrambled support vector vaulted verification," *Wksp App. of Computer Vision (WACV)*, 2012.

[17] G. Zhang F. Zheng and Z. Song, "Comparison of different implementations of mfcc," *Journal of Computer Science andTechnology*, vol. 16, pp. 582–589, 2001.

[18] B.-H. Juang and L. R. Rabiner, *Automatic speech recognition A brief history of the technology development*, Elsevier Encyclopedia of Language and Linguistics, second edition, 2005.

[19] HW Hon K. LEE and MY Hwang, "The sphinx speech recognition system," *Acoustics, Speech, and signal Processing (ICASSP-89)*, 1989.

[20] Mike Just and David Aspinall, "Personal choice and challenge questions: A security and usability assessment," *Proc. 5th Symposium on Usable Privacy and Security, 8*, 2009.

[21] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid, "Recommendation for key management part 1: General," *NIST special publication*, vol. 800, pp. 147, 2012.