

Authentication and Trust in Service-Centric Networking

Imad Aad*

Swisscom
Switzerland

imad.aad@swisscom.com

Torsten Braun

University of Bern
Switzerland

braun@iam.unibe.ch

Dima Mansour*

University of Basel
Switzerland

dima.mansour@unibas.ch

Abstract—Service-Centric Networking (SCN) extends Information-Centric Networking (ICN) from *content* to *services*, enabling clients to request services without having them coupled to specific servers. This brings new authentication challenges, since legacy authentication techniques do not apply. We propose and evaluate three generic SCN use cases and their corresponding authentication methods.

I. INTRODUCTION

With ICN, authenticating content can be done using public key cryptography: A content owner (e.g., a movie company) signs the content (e.g., a movie) using its private key. The content can then be distributed, cached, and retrieved. A user can check its authenticity using the public key of the content owner.

Service-Centric Networking (SCN) [2] is a more general paradigm that extends “content” in ICN to “services”, e.g., video transcoding or translation service. In Fig. 1, client C_1 requests a given service, say Translate-English-to-German(Hello). The request reaches server S , which is translation-capable and gets processed. The result is sent back and cached on the path from S to C_1 . If a similar request issued later on by another node passes via nodes J (ICN- and SCN-capable) or R (only ICN-capable), it can be served directly from their caches. On the underlying ICN network, SCN requests and replies are interpreted and treated as “content”, while at SCN nodes requests and replies are passed to the higher layer and “serviced” there.

A SCN example scenario is when a user requests a movie to be *transcoded* to be displayed on a mobile device. The movie is signed by the owner. Applying a transcoding service by a third party invalidates the signature, since the content has been changed. A solution could be that the content owner preemptively transcodes the movie to all possible encoding versions and signs them. This is non-practical, increases the overhead at the server, and reduces the advantages of using ICN/SCN. In this paper we explore possible solutions for authentication in three SCN use cases with increasing complexities.

II. AUTHENTICATION IN SCN

Since transcoding can be performed by SCN servers that are not necessarily trusted by the user, the user has no

* Work was done while being at University of Bern.

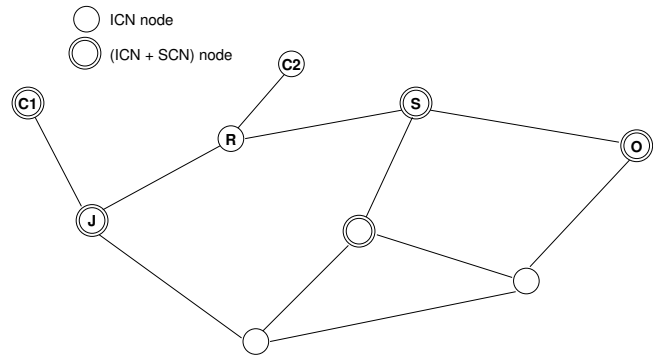


Fig. 1. ICN+SCN network example

guarantees that the content has not been tampered, potentially including indecent content, advertisements etc. We consider three generic use cases with increasing complexity levels:

- Converting temperature values (i.e., simple arithmetic operations): The owner publishes temperature values in °C, but the user prefers °F relying on a conversion service.
- Extracting specific content parts: A city authority publishes a list of events, attractions, restaurants. A user is interested in a subset of those, e.g., restaurants. Those are either requested from or advertised by a local server, but still need to be authenticated by the city authority.
- Transcoding images, movies: A content owner publishes a HD movie, but the user needs a transcoded version to be displayed on a mobile device with lower resolution.

III. SYSTEM MODEL

We describe stakeholders and trust assumptions as follows.

a) *Stakeholders*: Our system consists of *clients*, *ICN nodes*, *(ICN+SCN) service nodes*, where the SCN servers run, and *content owners* (ref. to Fig. 1). Clients C_i request some content to be “serviced” (i.e., transformed) by any SCN server S_i . The content can be fetched, from owner O or from any cache, by any SCN server or by the clients. ICN nodes (e.g., CCNx routers) forward requests and cache returned content, but do not interpret SCN messages. SCN servers do interpret SCN messages (e.g., requests) and can potentially execute given services requested by the SCN clients.

b) *Trust Assumptions*: Clients C_i and SCN servers S_i fully trust that content owners O are providing the appropriate content (to clients, routers, or servers). However, clients do not have any trust in service nodes S . Those can apply the requested service (e.g., transcoding) to the requested content (e.g., video), but some may maliciously inject unwanted content there (e.g., advertisements or indecent content). However, malicious SCN servers are not able to forge signatures of the content owner. The content owner is not involved in any of the requested services. Instead, it is required to *provide the content* and signing it using the methods introduced later.

c) *Cryptographic components*:

- *Content Owner* has a secret number (key) d for signing, and a public number (key) e for authenticating.
- $d \cdot e = 1$
- *Content Owner* has content a to publish, signs it by applying a^d , then publishes both.
- *Client* gets $\hat{a}$ (a possibly manipulated a) and a^d , raises this last to $(a^d)^e = a^{d \cdot e} = a$. If $a = \hat{a}$ then it knows that $\hat{a}$ has not been altered and that it is authentic.
- It is often more efficient to sign a hash of a , $H(a)$, than signing a . Verification is done by checking whether $(H(a)^d)^e = H(\hat{a})$.

Last, we assume that the private/public key scheme is unbreakable. Hence, the signatures cannot be forged by any malicious entity. The content owner uses the secret key for *signing* content a , denoted as $S(a)$. A client can use the owner's public key to *encrypt* a message m , denoted as $E(m)$.

IV. AUTHENTICATING SIMPLE ARITHMETIC OPERATIONS

Suppose a simple service takes a parameter onto which it applies an arithmetic operation, e.g., a service that transforms from Celsius to Kelvin: A content owner publishes the temperature in $^{\circ}C$, while the client prefers $^{\circ}K$, delegating the conversion ($^{\circ}K = ^{\circ}C + 273.15$) to an untrusted SCN server. Homomorphic cryptosystems allow users and servers to perform specific arithmetic operations on encrypted content without being able to decrypt it. We propose to make use of Paillier's cryptosystem [6] where messages m_1 and m_2 can be encrypted/signed as $E(m_1)$ and $E(m_2)$, and summing $E(m_1) + E(m_2)$ results in $E(m_1 + m_2)$:

- The content owner signs the value 273.15, i.e., $S(273.15)$, and publishes it.
- The content owner signs the temperature value $^{\circ}C$, and publishes both temperature and its signature $S(^{\circ}C)$.
- A SCN server retrieves the current temperature $^{\circ}C$ and computes $^{\circ}K = ^{\circ}C + 273.15$, then sends it to the client.
- The SCN server computes $S(^{\circ}C + 273.15) = S(^{\circ}C) + S(273.15)$ and sends it to the client.
- The client gets $^{\circ}K = ^{\circ}C + 273.15$ from the SCN server.
- Using the owner's public key, the client verifies whether $S(^{\circ}C + 273.15)$ corresponds to the received value $^{\circ}K = ^{\circ}C + 273.15$.

Paillier's signature scheme [6] uses the hash $h(m)$ of the content m instead of the content:

$$s_1 = \frac{L(h(m)^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} \bmod n \quad (1)$$

$L(u) = (u-1)/n$. To preserve the homomorphic properties in signatures, we require using the content itself (e.g., temperature value) rather than its hash. Converting from $^{\circ}C$ to $^{\circ}F$ requires multiplicative homomorphic properties such provided by ElGamal cryptosystem [3], where $E(m_1) \times E(m_2)$ results in $E(m_1 \times m_2)$. This enables the client to verify $^{\circ}C \times 9/5 = S(^{\circ}C) \times S(9/5)$ only, since the scheme cannot be combined with Paillier's to further verify the $+32$ ($S(^{\circ}C) \times 9/5 + 32 \neq S_{ElG}(^{\circ}C \times S_{ElG}(9/5) + S_{Paill}(32))$). This can be made possible if the content owner additionally signs $(^{\circ}C \times 9/5)$ using Paillier, which contradicts our assumption that the owner must not be involved in individual transformations.

V. AUTHENTICATING MODIFIED LISTS

In this case the owner publishes signed content and would like *any* other entities such as SCN servers to be able to take excerpts of the content while keeping the original signature valid. An example scenario can be a city authority publishing a huge list of events, attractions, restaurants etc., while a user is interested in a subset of these, e.g., restaurants. These are either requested from or advertised from a local server, but still need to be authenticated as signed by the city authority. A news agency might need to hide specific content, e.g., names of people, from an official document, while keeping the (official) signature of the original document verifiable by the end user. To support these applications we modify the *Sanitizable Signatures* scheme [1] and adapt it to our problem. The scheme in [1] considers a *signer*, a semi-trusted *sanitizer*, and a *verifier*. The signer splits a document m into t parts $m_1, \dots, m_t$, out of which k parts $m_{i_1}, \dots, m_{i_k}$ can be modified by the sanitizer, while the other $t - k$ parts must remain unchanged. Note that k is the maximum number of modifiable items in the list. The signer knows about the sanitizer prior to signing and sanitizing. The signer inputs the public key pk_{sanit} of the sanitizer to the signature, such that:

$$\sigma = s_{sk_{sign}}(ID_m || t || pk_{sanit} || \bar{m}_1 || \dots || \bar{m}_t) \quad (2)$$

ID_m is a document ID, introduced to avoid re-use of sanitization components for other documents, $s_{sk_{sign}}$ can be any standard signature scheme with any hash-and-encode mechanisms, such as RSA-EMSA-PSS. $\bar{m}_i = CH_{pk_{sanit}}(ID_m || i || m_i, r_i)$ for the sanitizable m_i , else $\bar{m}_i = m_i || i$ for the static parts of the document. r_i is the concatenation of all random coins r_{i_k} that are assigned to the sanitizable m_i of the document. $CH_{pk_{sanit}}(\cdot)$ is the *Chameleon Hash* with the particularity that the owner of the private key sk corresponding to the public key pk can find collisions, e.g., $CH_{pk}(m, r) = CH_{pk}(m', r')$ for $m \neq m'$, and $r \neq r'$. Therefore, a sanitizer can modify parts of the document m_i into m'_i , while keeping the original signature valid, i.e.:

$$\sigma = CH_{pk_{sanit}}(ID_m || i || m_i, r_i) = CH_{pk_{sanit}}(ID_m || i || m'_i, r'_i) \quad (3)$$

The verifier needs σ , m , r , and additional information about how the document was split to check the signature's validity. This differs from [1] as follows:

- The modified part is only a deletion, which can be a replacement by a special sequence of characters (e.g. “/////”), instead of a modification to any new content. i.e., $m'_i = “/////”$ for all i of the sanitizable content parts.
- It requires that *anyone* (e.g., untrusted SCN server) is able to modify the content parts, more precisely to remove specific parts of the document while keeping the original signature valid. In [1] the public key of the sanitizer pk_{sanit} is needed as input to the signature for non-repudiation, which we do not need in our scenario.

Therefore, we replace the public key of the sanitizer pk_{sanit} by the one of the signer pk_{sign} in Eq. (2) which also computes the r_i and r'_i such that $m'_i = “/////”$ for all sanitizable m_i , and that Eq. (3) is preserved, then publishes the r (together with σ) for usage on the verifier's side.

VI. AUTHENTICATING MULTIMEDIA CONTENT

Watermarking protects the ownership of digital images, movies, and their integrity. Watermarking techniques [4], [5] aim at protecting databases against typical database alterations or attacks such as tuple insertion, deletion, and value modification, without altering the data due to watermarking. We retain the following common issues:

Where to put the watermark? In SCN the data owner cannot hide the watermark in the data for integrity or ownership checks, since it must be easily accessible to the user for verification. Alternatively, the user could request the watermark location from the owner, involving the owner of the data upon each usage, which goes against the principles of SCN/ICN.

How to protect? Using public/private key signatures is a common technique and also applicable to SCN authentication.

What to protect? Signing the entire multimedia file would be invalidated whenever the file gets modified by any service. We identify the need to protect the multimedia *content* rather than the multimedia *file*, and more precisely only the values that are robust to transcoding.

A. Authenticating transformed images

Consider the example of an image scaled down using SCN. Protecting (i.e., the owner signing) a subset of the image pixels would be invalidated even by a well behaving service provider scaling down the image, since some of those pixels may be averaged or removed. However, for the specific scaling function other image properties such as average value of all image pixels, average values of rows or columns, or FFT of the image remain the same.

Signing the average value of all pixels can be the approach with least computation and communication overhead, but it is the least secure: A malicious service provider can replace the image by an image with the same average value, thus keeping the owner's signature valid, and fooling the end user.

Averaging the values of rows/columns of the image, then signing the resulting vector(s) adds another level of complexity

to the malicious service provider. Changing the content is still possible, if the attacker tweaks some rows/columns to make their averages equal to the ones of the signed image.

Signing the FFT of an image introduces an additional layer of security which protects, not only the values of the image pixels, but also the relationships between them. An attacker would have to forge an image with similar pixel-relationship to the original image to fool the end user. This technique surely improves the robustness of the signing scheme considerably. However, it also increases communication overhead, because the owner must publish the image and the signed FFT of the image, and computation overhead, because the end user must compute the FFT of the received image and compare it with the signed FFT of the original image. We do not require any meta-data to be issued by the content owner, beyond the content itself and its corresponding signature(s). Therefore, we assume the client must perform such operations again on the content it receives. Averaging pixels/rows/columns or applying FFTs to the original size image and then to the scaled ones, may result in slightly different values. Therefore, the owner's signature must be applied to *ranges* of acceptable values rather than on strict unique values. The proposed technique for authenticating transformed images is not secure in the cryptographic sense. A malicious server can subtract/add values to the pixels in order to match the signed averages. Our proposed technique simply makes the malicious server's task harder to produce a *visually valid image*.

B. Authenticating transformed videos

Transcoding can change bit rate or movie resolution. If changing the bit rates, transformations are changing quantization, while keeping resolution, number of frames, and motion vectors unchanged. If changing the movie resolution, transformations are scaling I-frames and motion vectors. Therefore, our proposed techniques for image authentication can still be applied to I-frames, to which we can add the motion vectors before signing. In bit-rate transformation, average values should remain the same, while in resolution transformation image and motions vectors are scaled, therefore scaling the averages proportionally. Consider the case of signing still images for authentication after being “serviced”, e.g., being scaled to half and quarter of the original size (cf. Table I). We consider 14 different pictures. Some of them have content similarities, indicated in the “simil” column. Those pictures were shot within very short time intervals. We need to sign attributes that remain (almost) unchanged before and after applying the service, such as average value of all image pixels, average values of individual rows and/or columns, or FFT of the image. Table I shows the average value of all image pixels. The average values change slightly (approx. 0.1%) when the image is scaled, which makes it a valid attribute to sign for authentication.

However, since this average is changing, even slightly, the content owner must sign the *range of values* indicated by [min;max] in the table, instead of a fixed single average value.

Image	Simil.	Size	Average					
			full	half	quarter	[min ;	max]	%
img-01		1224x1632	89.504	89.591	89.484	89.484	89.591	0.12
img-02		3888x2592	76.060	76.187	76.096	76.060	76.187	0.17
img-03	*	1536x2048	94.359	94.479	94.378	94.359	94.479	0.13
img-04	*	1536x2048	99.497	99.602	99.501	99.497	99.602	0.11
img-05	*	1536x2048	101.931	102.028	101.931	101.931	102.028	0.10
img-06		1536x2048	94.849	94.945	94.840	94.840	94.945	0.11
img-07		2048x1536	84.893	85.002	84.903	84.893	85.002	0.13
img-08		2048x1536	91.575	91.700	91.601	91.575	91.700	0.14
img-09		2048x1536	86.754	86.883	86.792	86.754	86.883	0.15
img-10		2048x1536	107.959	108.076	107.975	107.959	108.076	0.11
img-11		1536x2048	88.162	88.282	88.182	88.162	88.282	0.14
img-12	"	640x480	143.531	143.635	143.532	143.531	143.635	0.07
img-13	"	640x480	142.171	142.274	142.179	142.171	142.274	0.07
img-14	"	640x480	149.933	150.052	149.948	149.933	150.052	0.08

TABLE I
AVERAGING RGB PIXELS OF ORIGINAL IMAGES AND THEIR SCALED VERSIONS

Therefore, the owner must publish the content (i.e., image) together with the [min;max] margin signed.

Upon receiving the “serviced” image (i.e., scaled), the client computes the average pixel value. If it falls within the [min;max] margin and the signature of the margin is valid, the client can infer that the “serviced” image is valid. To determine these margins and then sign them, the owner must perform all possible transformations on its images, which contradicts the goal of SCN: An owner publishes a single content object, which is then serviced “*elsewhere*”. To overcome this drawback, the column (%) in Table I indicates that those margins have almost the same proportional width. Therefore, the owner can specify a single average value (of the original size image) together with this proportional margin width, and sign them. The [min;max] columns in Table I suggest that these margins do not overlap, even for images with high similarities. This may mislead into thinking that images cannot be replaced by others and that these margins of averages can be used for reliable signatures. In fact, a malicious SCN server still can:

- Have a large database of images (beyond 14) that cover a wide range of average pixel values. Receiving a picture P_1 to scale, and a signed margin $[P1_{min}; P1_{max}]$, the malicious server can choose a picture P_2 from his database, the average of which falls in $[P1_{min}; P1_{max}]$, and send it to the client, which would (mistakenly) verify that P_2 is authentically from the owner.
- Pick any picture P_3 and forge it by adding a few lines, columns (e.g., a frame around it), to make its average fall within $[P1_{min}; P1_{max}]$.

This leads us to the other two attributes of images that not only remain (almost) constant over transformations, but are also robust to forging by malicious SCN servers: averages over pixels of rows and/or columns, and FFTs of the pictures. When we run the same pixel averaging as above, but on individual rows and/or columns, we get similar observations: columns and rows maintain similar averages before and after transformation. However, it results in thousands of margins at

the output, instead of one, that can be signed all together and published by the owner. Therefore:

- The owner produces a vector or margins, typically of thousands of elements.
- The owner must perform relatively more processing to compute it, which can be done offline once.
- Signing it does not bring any major difference.
- The vector of margins is clearly larger than a single margin of two values. To reduce the communication overhead the owner can compress it.
- The client has to perform considerably more computations to average thousands of rows/columns to verify the validity of the “serviced” content received.

The major difference in using this attribute is at the side of a malicious SCN server: To find a fake image in its database or to forge a specific one such that the signature is still considered as valid by the client, the malicious SCN server has to make sure that the averages of *all* rows and columns of the fake image fall within *all* the margins published and signed by the owner, which is a tremendous task. Similarly, FFTs (slightly) increase the owner’s and client’s computation, but make the forging task of the malicious server practically infeasible.

REFERENCES

- [1] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik, “Sanitizable signatures,” in *ESORICS*, 2005, pp. 159–177.
- [2] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner, and M. Varvello, “Service-centric networking,” in *IEEE International Conference on Communications Workshops (ICC)*, 2011.
- [3] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in *IEEE TIT*, 1985.
- [4] R. Halder and A. Cortesi, “A persistent public watermarking of relational databases,” in *ICISS*, 2010.
- [5] Y. Li and R. H. Deng, “Publicly verifiable ownership protection for relational databases,” in *Proceedings of ASIACCS*, 2006.
- [6] P. Paillier, “Public-key cryptosystems based on composite degree residuosity class,” in *EUROCRYPT*, 1999.