

## **Historiographical Study of Authentication Approaches in Computer Systems**

PhD candidate Petar Dimitrov  
University of Economics - Varna, Varna, Bulgaria  
p.d.dimitrov@ue-varna.bg

Prof. DSc Pavel Petrov  
University of Economics - Varna, Varna, Bulgaria  
petrov@ue-varna.bg

### **Abstract**

*The study focuses on the history of authentication approaches in computer systems in order to reveal the evolution of computer security. The main historical stages in authentication approaches in computer systems are related both to the development of computers and networks themselves, and to the growing need for a higher level of security and protection of personal data. Technological and user's behavioral risks have led to the gradual replacement of the password by more dynamic, context-based methods, through which an attempt is made to overcome the limitations of human memory and the growing threats in modern information systems interconnected by computer networks. The large number of accounts that modern users use to access multiple information systems, as well as the need for unique, complex passwords, lead to the so-called "password fatigue" phenomenon, in which users start using the same or easy-to-remember passwords, which significantly lowers the level of computer security. This leads to the emergence of many new and inherently different authentication methods that various organizations are trying to impose on the market.*

*Keywords: password, authentication, hashing, cryptography, security.*

*JEL Code: L86*

*DOI: 10.56065/IJUSV-ESS/2025.14.2.126*

### **Въведение**

В контекста на информационните технологии паролите започват да се използват масово след създаването на компютри, работещи в режим на разделение по време (time-sharing) в началото на 60-те години на XX век (Corbato et al., 1962). По него време в Масачузетския технологичен институт (MIT) е създадена Compatible Time-Sharing System (CTTS), по който проект работи екипът на Фернандо Корбато. Основният проблем, пред който са били изправени Корбато и колегите му, е как да предоставят на множество потребители едновременен достъп до различни файлове и ресурси, без да се нарушава сигурността и неприкосновеността на информацията. Имало е идеи за алтернативни методи на удостоверяване чрез въвеждане на отговор на предварително зададен въпрос, но системата с персонални пароли се оказва по-надеждна и лесна за управление, което я прави предпочитаната опция и се използва в първите мейнфрейм системи (Corbato et al., 1962; Saltzer & Schroeder, 1975).

В днешно време използването на пароли е свързано с множество проблеми. Например, множество потребители използват кратки и лесни за запомняне пароли, което ги прави уязвими към атаки от тип brute-force, dictionary и rainbow таблици (Vonneau et al., 2012). Освен това, в съвременните веб-базирани платформи и мобилни приложения се наблюдава тенденция към нарастване на броя на акаунтите на един потребител, което води до повторно използване на пароли, което от своя страна, увеличава риска от компрометиране на данни при изтичане на информация от една услуга (Florencio & Herley, 2007).

За решаването на тези проблеми, се извършва преход към многофакторна автентикация, която комбинира парола с други форми на удостоверяване, като хардуерни токени, мобилни приложения за генериране на еднократни пароли или биометрични данни (Aloul et al., 2009). В този контекст се появява стандартът U2F (Universal 2nd Factor) и по-късно

WebAuthn, които осигуряват по-високо ниво на сигурност (Petrov et al., 2020), елиминирайки слабостите на традиционните пароли и минимизирайки риска от фишинг и други онлайн атаки. Така, историческото развитие на паролите от MIT до съвременните решения като U2F и WebAuthn демонстрира еволюцията на автентикационните методи - от просто средство за идентификация до интегрирана система за сигурна цифрова идентичност, която съчетава удобство, защита на лични данни и защита от конвенционални атаки (FIDO Alliance, 2023).

### **1. Начини за съхранение на данните за автентикация**

Съвременните системи за управление на достъпа изискват надеждни методи за съхранение на автентикационните данни. Данните за автентикация, включително потребителските имена и паролите, могат да бъдат съхранявани по различни начини, всеки от които има свои предимства и недостатъци. В миналото подходът е бил простото съхраняване на автентикационните данни в текстови файлове или бази данни в явен вид<sup>1</sup> (plain text), но в последствие се въвеждат хеширане, посоляване (salting) и криптиране като стандартни методи<sup>2</sup> за защита на чувствителната информация (Stallings, 2017; Bonneau et al., 2012).

Макар криптографията като наука да се развива от хилядолетия, нейното развитие се засилва в началото на XX век с развитието на механичните и електромеханичните сметачни машини. Особено силно влияние оказват и проведените световни войни, където войващите страни използват шифри при комуникация. През 1972 г. Националното бюро по стандартизация (National Bureau of Standards, NBS) в САЩ създава стандарт за криптиране на неklasифицирана и чувствителна информация, който да бъде разпространен сред всички правителствени структури. Усилията на NBS, да се намери подходящ алгоритъм, са съвместни с Националната агенция по сигурността (National Security Agency, NSA) на САЩ и през 1974 г. се намира алгоритъм, който да отговаря на изискванията. Този алгоритъм е предложен от компанията IBM и е бил изграден на основата на алгоритъма, разработен от Хорст Файстел, наречен Луцифер. Той представлява една от първите имплементации на симетрично криптиране, т.е. алгоритми, използващи един и същ криптографски ключ както за криптиране, така и за декриптиране на информацията. Алгоритъмът на IBM, който впоследствие се превръща в федерален стандарт за обработка на информация Data Encryption Standard (DES), използва 56-битов критиращ ключ<sup>3</sup>.

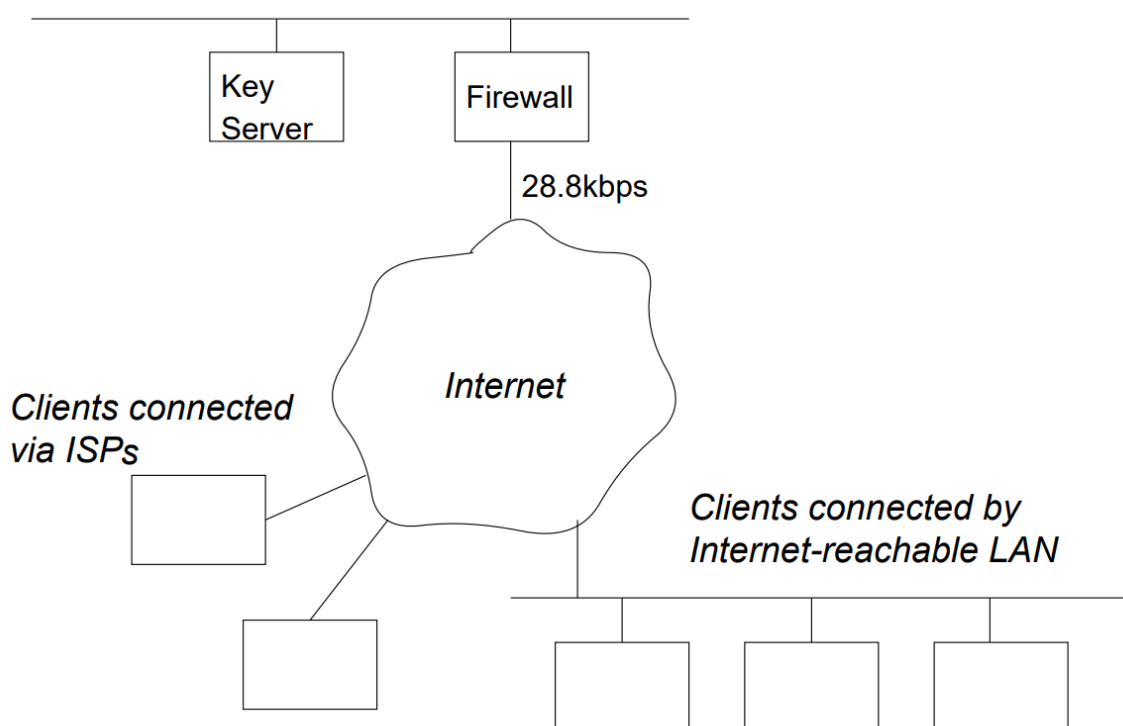
---

<sup>1</sup> В първите компютърни системи, като CTSS на MIT, автентикационните данни често се съхраняват във файлове в текстов вид. Например, идентификаторите на потребителите и техните пароли са били записвани в файл с име UACCNT.SECRET, който файл е имал права за четене от някои от потребителите на системата (Corbato et al., 1962). Алан Шер, докторант в MIT, установява, че чрез функционалността на системата за разпечатване на файлове е възможно да се получи достъп до паролите на други потребители. По-късно, поради човешка грешка, файлът с пароли е разменен с файлът MotD (Message of the Day) и по този начин всички потребители са получили достъп до идентификаторите и паролите на останалите потребители (Saltzer & Schroeder, 1975).

<sup>2</sup> Въпреки, че подходът за съхранение в текстов вид днес се счита за несигурен, практиката на съхранение на пароли в plain text все още може да се срещне в съвременните системи, понякога поради технически или организационни пропуски. След задълбочен анализ и проучване за възможните начини за справяне с проблема с лесно компрометирана система, учените още през 60-те години решават да се използват криптографията като метод за повишаване сигурността на паролите.

<sup>3</sup> DES има ефективен размер на ключа от 56 бита, т.е. повече от 72 квадрилона възможни ключа. Въпреки че това число изглежда огромно, съвременните компютърни системи и специализиран хардуер (като например DES-Cracker) могат да изпробват всички тези комбинации за сравнително кратко време, поради което DES вече не се счита за сигурен за общо ползване. Още по времето на неговото първоначално внедряване в различни системи Мартин Хелман и Уитфилд Дифи от Станфордския изчисляват, че 72 квадрилона комбинации могат да бъдат обработени за един ден от машина, струваща 20 милиона щатски долара - сума, която е в рамките на бюджетите на различни държавни институции. Поради тези опасения на 28 януари 1997 RSA Security организира серия от „състезания“ с награден фонд в размер на 10000\$, които да докажат невъзможността на DES алгоритъма да пази чувствителни данни.

Един от първите мащабни опити за декриптиране на тестово съобщение на RSA Security<sup>4</sup>, проведено от група учени, водени от Роки Върсър, е DESCALL Project - съкратено от DES Challenge (фиг.1). Върсър разработва концептуален модел за използването на множество компютри, свързани в мрежа като обявява този модел в Usenet през 1997 г. Сървърът, който координира действията на множеството компютри е бил базиран на 486 архитектура и е имал 56MB оперативна памет. Разработен е приложен софтуер, който трябва да бъде използван на компютрите на доброволците, като дори се създава и софтуер за по-съвременните x64 архитектури. С помощта на този софтуер, на обикновен по него време персонален компютър, притежаващ процесор Pentium с работна честота 200 MHz, е можело да се изпробват около 1 милион ключа в секунда. Броят на компютрите, използващи този софтуер към края на проекта надхвърля 14 хиляди за денонощие. По този начин скоростите достигат до 7 милиарда ключа в секунда. Търсеният ключ е намерен 56 часа след стартирането на проекта, като за това време са обработени около 25% от всички възможни ключове<sup>5</sup>.



Фигура 1. Архитектура на DESCALL Project

Източник: Curtin & Dolske, 1998

Авторите на проекта стигат до няколко ключови извода: системи, в които ключът е кратък не предоставят нужното ниво на сигурност; мощностите, нужни за провеждане на декриптиране стават все по-достъпни; възможността за изпълнение на огромни изчислителни задачи без помощта на суперкомпютри или скъп хардуер чрез средствата на Internet-свързаността. Проектът DESCALL успява да докаже на обществеността, че с достатъчно изчислителна мощ за 96 дни DES алгоритъма за криптиране може да бъде преодолян. Скоро

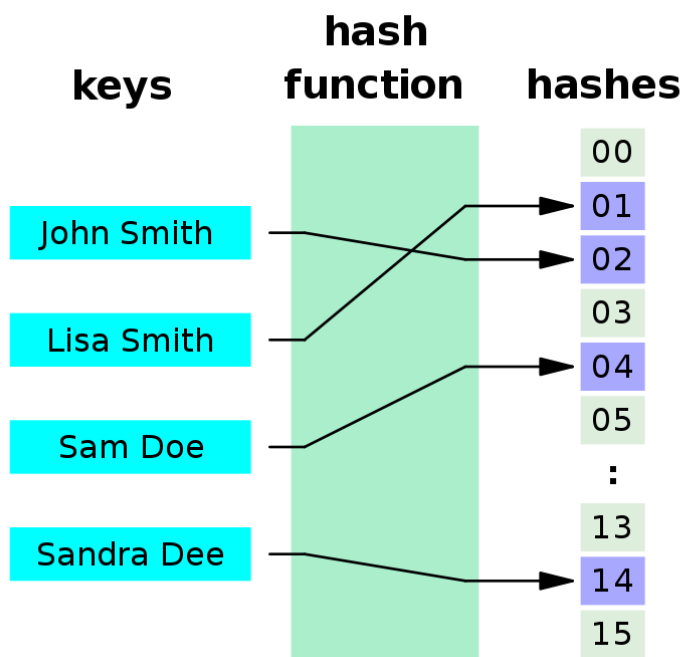
<sup>4</sup> RSA Security LLC е американска компания за киберсигурност, основана през 1982 г. от създателите на RSA алгоритъма: Рон Ривест, Ади Шамир и Леонард Адълман

<sup>5</sup> Собственикът на компютъра, който намира ключа е получил \$4000, а останалите средства се разпределят сред създателите на софтуерния проект.

след първото успешно декриптиране, времето нужно за декриптиране значително намалява от 96 дни на 22 часа<sup>6</sup>.

След като времето за декриптиране пада под 24 часа става ясно, че DES вече не притежава нужните качества да защити една информация. В контекста на слабостите на симетричните алгоритми като DES и нарастващите изисквания към защитата на данните, изследователската общност започва да обръща внимание и на други криптографски механизми, които да осигурят по-висока степен на сигурност. Един от тези механизми е хеширането – техника, която се различава от класическото шифриране, но играе ключова роля за защитата на пароли, цифрови подписи и удостоверяване на данни. Значимостта на хеширането като еднопосочна функция е изведена още в класическите трудове на Дифи и Хелман (Diffie & Hellman, 1976)<sup>7</sup>, а по-късно е разгледана като фундаментален компонент на съвременната криптография от Menezes, van Oorschot и Vanstone (1997). Терминът хеширане (от англ. каша, нарязвам) за пръв път се намира в научната литература още през 60-те години на XX век, макар да е широко разпространен термин и преди това.

При хеширащите функции обикновено входните данни са по-големи от изходните и това им свойство предполага наличието на т.нар. колизии. Това се обуславя и от принципа на Дирихле, според който, ако имаме  $m$  на брой предмета и  $n$  на брой клетки, ако  $m > n$ , то поне една от клетките съдържа  $m:n$  предмета, т.е. в една клетка има повече от един предмет. Разбира се има теоретични обосновки за хешираща функция без колизии, която извежда уникален хеш за всеки различен низ от данни, който ѝ се подаде на входа (фиг.2).



Фигура 2. Идеална хешираща функция за четири имена

Източник: Jorge Stolfi, Wikimedia, 2009

<sup>6</sup> Състезания, организирани от RSA Security и време за декриптирането им: в състезанието DES Challenge I, проведено през 1997 г. печели DESCHALL Project - 96 дни; в състезанието DES Challenge II-1, проведено през 1998 г. печели distributed.net - 39 дни; в състезанието DES Challenge II-2 проведено през 1998 г. печели Deep Crack - 56 часа; в състезанието DES Challenge II-3 проведено през 1999 г. печели distributed.net, Deep Crack - 22 часа. Източник: RSA Data Security. RSA Laboratories Secret-Key Challenge, <http://www.rsa.com/rsalabs/97challenge/>

<sup>7</sup> През 1976 г. Уитфилд Дифи и Мартин Хелман идентифицират нуждата от скриване на чувствителни данни (най-често пароли) чрез еднопосочна функция, чиято задача е да обработи входни данни с различна дължина и да изведе данни със строго зададен размер и формат.

Идеалната хешираща функция може да се опише със следните критерии: да има детерминистичен характер<sup>8</sup>; да може да изчисли хеш за всякакви входни данни бързо; да бъде невъзможно генерирането на входни данни, които извеждат конкретен хеш; да бъде невъзможно намирането на две различни входни данни, които да генерират един и същ хеш (устойчивост за колизии); малка промяна във входните данни трябва да генерира хеш, който да не може да бъде сравняван с хеш, генериран от данните преди промяната<sup>9</sup>.

Според Menezes, van Oorschot и Vanstone всяка една хешираща функция се състои от два основни компонента - компресираща функция и конструкция. Компресираща функция превръща даден низ от данни в по-кратък такъв, а конструкцията представя начина, по който компресиращата функция се извиква в процеса на хеширане до извеждане на крайния резултат (Menezes, van Oorschot, & Vanstone, 1997). Също според горепосочените автори хеширащите функции се категоризират в три различни класа - функции, базирани на блок-шифри, базирани на модулна аритметика и на специализирани хеширащи функции (пак там). Първите опити на учените да създадат хеширащи функции са базирани на блок-шифрите и по-конкретно тези на DES алгоритъма за криптиране. Един от големите проблеми при този подход е невъзможността за извеждане на резултат с дължина, по-голяма от тази на блока - 64 бита в случая с DES. Това неминуемо води до генериране на колизии.

Други автори се обръщат към модулната аритметика, за да създадат модели на хеширащи функции като се основават на теоретични допускания като факторизация и дискретна логаритмизация. Като пример за такава функция може да се посочи труда на Михир Белар (Bellare, 1994), в който се предлага функцията да се базира на трудното изчисление на дискретната логаритмизация в комбинация с прости числа. Друг пример е VHS (Very Smooth Hash), създаден от Скот Контини, Аржен Лестра и Рон Стайнфелд през 2005 (Contini, 2006). Поради това, че функцията е базирана на проблем в математиката, който се смята за труден за решаване, намирането на колизии би било толкова трудно, колкото решаването на самия проблем. Само година по-късно Марку-Юхани Сааринен намира пропуски в VHS, с които доказва, че някои от свойствата, нужни за една хешираща функция липсват (Saarinen, 2006).

Многото ограничения при използването на блокови шифри довеждат до разработването на специализирани функции, оптимизирани за използване в потребителски софтуер. Един от най-типичните примери е MD5 алгоритъма, предложен от Роналд Риветс през 1991 г., като преди това той работи и по MD2, MD4 и др. Алгоритъмът MD5 (съкратено от Message Digest 5) е публикуван като усъвършенствана версия на MD4, като скоростта му на работа е десетократно по-бърз от DES, когато се използва в софтуерна среда. Характерна черта на MD5 е, че е достъпен без лицензиране и е лесен за експортиране в сравнение с хардуерно-базираните алгоритми. Тези му качества обуславят изключително високата му популярност и до днес, макар че още през 1992 г. се намират пропуски и възможност за генериране на колизии (den Boer & Bosselaers, 1993), дори някои за части от секундата (Stevens et al., 2009). Риветс използва пет стъпки за хеширането като първите подготвят низа като добавят битове в началото и в края, а в следващите се прилагат предварително дефинирани функции. Въпреки наличието на колизии обаче, алгоритъмът има едно друго широко разпространено приложение, а именно верификация на трансфер на данни от един сървър до друг или от сървър до потребител. Тази имплементация се осъществява като се изчисли MD5 хеша на оригиналните данни и след това се сравни с изчисления хеш на получените данни - ако те съвпадат, това означава, че данните са успешно прехвърлени в цялост. Подобно приложение се използва именно благодарение на бързината на алгоритъма.

---

<sup>8</sup> За да бъде една функция от детерминистичен характер тя трябва да извежда една и съща стойност при подаването на едни и същи входни данни.

<sup>9</sup> Ефектът на лавината за пръв път е използван като термин от Хорст Файстел (Feistel, 1973), но концептуално е разработен от Клод Шанон в неговия секретен, за времето си, труд „Математическа теория на криптографията“ през 1945 г.

Вече беше споменато, че хеширащата функция е детерминистична, т.е. винаги едни и същи входни данни генерират един и същ резултат. С това си поведение при евентуален пробив в сигурността и нерегламентиран достъп до списъка с хеширани пароли, потребители с едни и същи пароли биха имали едни и същи хешове. В своя труд от 1978, озаглавен „Password Security: A Case History“, Кен Томпсън нарича 12-битов низ, генериран на база часовника на компютъра, “salt” (от англ. сол) (Thompson & Morris, 1979). Този низ от случайно генерирани символи се прибавя към въведената от потребителя парола и получения низ се изпраща към хеширащата функция. По този начин потребители с еднакви пароли притежават различна хеширана парола и различен salt низ. Типичен пример за използване на salt низ е файлът /etc/passwd (фиг. 3), който е характерен за ранни версии на Unix базирани системи до 80-те години. В него се съхранява както потребителското име, потребителската група, така и самия хеш на паролата и salt низа (в некриптиран вид). Всички потребители в системата имат достъп до този файл, тъй като множество от системните приложения го използват, като по този начин единствената защита на потребителските пароли е еднопосочното им хеширане. В тези по-стари версии на Unix дължината на salt низа е била ограничена до 12 бита (4096 възможни комбинации) и тъй като изчислителните мощности продължават растежа си, бързо става ясно, че е нужна допълнителна защита.

```
# cat /etc/passwd
root:x:0:0:ROOT account:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
myuser:x:513:520:Test User:/home/myuser:/bin/bash
----- output truncated -----
```

Фигура 3. Структура на /etc/passwd файл

*Източник: Собствена разработка*

От 80-те години насам във всички Unix дистрибуции се прилага използването на /etc/shadow (фиг. 4) файл, който е ограничен за достъп само до root потребителя на системата и по този начин позволява съхраняването на хеширани пароли и salt низове без възможността всеки потребител на системата да може да получи достъп до тях.

```
# cat /etc/shadow
root:$1$UFnkhP.mzcMyajdD9OEY1P80:17413:0:99999:7:::
bin:!:15069:0:99999:7:::
daemon:!:15069:0:99999:7:::
adm:!:15069:0:99999:7:::
testuser:$1$FrWa$ZCMQ5zpEG61e/wI45N8Zw.:17413:0:33:7:::
```

Фигура 4. Структура на /etc/shadow файл

*Източник: Собствена разработка*

В проучване на Bonneau & Preibusch използването на salt низ в никакъв случай не прави паролата на даден потребител неразбиваема (Bonneau & Preibusch, 2010; NIST, 2017). Нещо повече - ако хакер има достъп до хеш на парола и salt низ за конкретен потребител, то единствената трудност, който salt низа добавя е логиката, стояща зад хеширащата функция (напр. дали salt низа е добавен преди или след паролата на потребителя). Основната причина използването на salt низ да е все препоръчвана добра практика е заради възможността за добавяне на ентропия към хешираните пароли на потребителите.

Възможно е да се възприеме идеята, че при повторно хеширане на дадена парола, сигурността се увеличава, но това твърдение не винаги е вярно. Теорията на вероятностите и по-конкретно проблема с рождената дата твърди, че в един клас от 30 ученика вероятността един от тях да има същия рожден ден като друг ученик е около 70%. Този подход към математиката е довел до разработване на т.нар. “Birthday” (пр. от англ. рожден ден) атака в криптоанализата. Нека вземем за пример MD5 като алгоритъм за хеширане. Макар да е известен в криптографските среди като алгоритъм, при който колизии са възможни, ако приемем теоретично, че шансът за образуване на колизия е 0.001% за конкретен брой записи, и изпълним хеширане с MD5 върху конкатенираните парола и salt-низ, то при повторно хеширане на получения резултат с MD5, бихме увеличили вероятността на колизия двойно. За да се предпазим от подобен ефект е нужно към получения хеш от първа стъпка да се конкатенира salt-низата отново преди да бъде хеширан.

Друг широко разпространен метод за съхранение на чувствителни данни като потребителски пароли е използването на техника, наречена „key stretching” (от англ. удължаване на ключа). Първото документирано използване на подобен метод е описан в труда на Робърт Морис през 1978. Заедно с Кен Томпсън използват 25 итерации DES алгоритъм за хеширане като за входни данни се използват 12-битови salt низове. Дължината на паролите е имала ограничение от 8 ASCII символа и макар използваните тогава итерации и максималната дължина на паролите и на salt низа да позволяват на един съвременен компютър да декриптира парола изключително бързо, съвременни производни функции като например PBKDF2 (Password-Based Key Derivation Function 2), използващи SHA-2 като алгоритъм на хеширане, по-голяма дължина на salt низа и повече итерации са доста ефективни в подобряване на сигурността на съхраняване на потребителските пароли.

### **Заклучение**

Направеният преглед за историческото развитие на паролите в компютърните системи показва, че първите пароли се появяват в ерата на големите мейнфрейм системи, когато няколко потребители споделят един и същ компютър чрез терминали през 1960–1970 г. Паролите се съхраняват в чист текст, което ги прави уязвими при достъп до системата, което води до т.нар. „изтичане на пароли“. През 1974 г. в UNIX се въвежда криптирано съхранение на пароли чрез функцията `crypt()`, първоначално в `/etc/passwd` файл, а по-късно към средата на 1980-те г. в „shadow password“ файл с ограничен достъп.

Развитието на локални мрежи (LAN) и клиент-сървър архитектурата налага централизирано управление на автентикация. MIT разработва Kerberos през 1988 г., базиран на симетрично криптиране и времеви „билети“, с цел предотвратяване на подслушване на пароли по мрежата. Microsoft и Novell внедряват мрежови домейни и централизираны политики при което паролите вече се управляват чрез домейн контролери и политики за дължина, сложност и изтичане.

В началото на ерата на уеб технологиите и масовото използване на Интернет в периода 1990–2000 г. паролите се превръщат в основен метод за удостоверяване на потребители онлайн. Един потребител започва средно да използва десетки пароли за достъп до различни системи - електронна поща, уебсайтове, онлайн банкиране (Petrov et al., 2017; Petrov et al., 2018; Petrov et al., 2019) и др., което води до повторно използване и слаби комбинации. В

отговор на нарастващите киберзаплахи и хакерски атаки (особено атаки с груба сила и речникови атаки), се въвеждат изисквания за по-сложни пароли (дължина, комбинация от главни/малки букви, цифри и специални символи). Големият брой акаунти и необходимостта от уникални, сложни пароли водят до т.нар. „паролна умора“ (password fatigue), при което потребителите започват да използват едни и същи или лесни за запомняне/отгатване пароли, което отново понижава сигурността. За справяне с паролната умора, популярност набират мениджърите на пароли (password managers), които генерират сложни и уникални пароли за всеки акаунт, които впоследствие съхраняват сигурно.

След 2005 г. се въвежда двуфакторна/многофакторна автентификация (2FA/MFA) – комбинация от парола и временен код чрез SMS или токен, и по този начин паролите вече са само един от многото фактори за удостоверяване в рамките на „контекстуален достъп“. След 2010 г. се прилагат нови подходи за безпаролно (passwordless) удостоверяване чрез хардуерни ключове (YubiKey), биометрия - пръстови отпечатащи, лицево разпознаване (Dimitrov et al., 2020) или криптографски токени. След 2020 г. с популярност се ползват FIDO2 и WebAuthn в резултат на засилване на нормативните регулации (например NIST SP 800-63B, GDPR, NIS2) относно управление на идентичности и защита на достъп.

Общата тенденция е, че чрез внедряване на нови методи се предлага по-добра сигурност и по-голямо удобство за потребителите.

### References

1. Aloul, F., Zahidi, S., & El-Hajj, W. (2009). Two factor authentication using mobile phones. In 2009 IEEE/ACS international conference on computer systems and applications, pp.641-644.
2. Bellare, M., Goldreich, O., & Goldwasser, S. (1994). Incremental cryptography: The case of hashing and signing. In Annual International Cryptology Conference, Berlin, Heidelberg: Springer Berlin Heidelberg, pp.216-233.
3. Bonneau, J., & Preibusch, S. (2010). The Password Thicket: Technical and Market Failures in Human Authentication on the Web. In Proceedings of the Ninth Workshop on the Economics of Information Security (WEIS 2010). [Online] Available from: [https://weis2010.econinfosec.org/papers/session3/weis2010\\_bonneau.pdf](https://weis2010.econinfosec.org/papers/session3/weis2010_bonneau.pdf) [Accessed 11/10/2025]
4. Bonneau, J., Herley, C., Van Oorschot, P. C., & Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In 2012 IEEE symposium on security and privacy, pp.553-567.
5. Contini, S., Lenstra, A. K., & Steinfeld, R. (2006). VSH, an efficient and provable collision-resistant hash function. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, Berlin, Heidelberg: Springer Berlin Heidelberg, pp.165-182.
6. Corbato, F. J., Merwin-Daggett, M., & Daley, R. C. (1962). An experimental time-sharing system. In Proceedings of the AFIPS Spring Joint Computer Conference, pp. 335-344.
7. Curtin, M. & Dolske, J. (1998). A Brute Force Search of DES Keyspace, ;login: USENIX magazine, [Online] Available from: <https://web.interhack.com/publications/des-key-crack.pdf> [Accessed 11/10/2025]
8. den Boer, B., & Bosselaers, A. (1993). Collisions for the compression function of MD5. In Workshop on the Theory and Application of Cryptographic Techniques, Berlin, Heidelberg: Springer Berlin Heidelberg, pp.293-304.
9. Diffie, W., & Hellman, M. (1976). New directions in cryptography. IEEE Transactions on Information Theory, 22(6), pp.644–654. DOI: <https://doi.org/10.1109/TIT.1976.1055638>
10. Dimitrov, G. P., Bychkov, O., Petrova, P., Merkulova, K., Zhabska, Y., Zaitseva, E., ... (2020). Creation of Biometric System of Identification by Facial Image. In 2020 3rd International Colloquium on Intelligent Grid Metrology (SMAGRIMET), IEEE. pp.29-34. DOI: <https://doi.org/10.23919/SMAGRIMET48809.2020.9263995>

11. FIDO Alliance. (2023). FIDO2: Moving the World Beyond Passwords. [Online] Available from: <https://fidoalliance.org/> [Accessed 11/10/2025]
12. Florencio, D., & Herley, C. (2007). A large-scale study of web password habits. In Proceedings of the 16th international conference on World Wide Web, pp.657-666.
13. Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1997). Handbook of applied cryptography. CRC press. DOI: <https://doi.org/10.1201/9781439821916>
14. Morris, R., & Thompson, K. (1979). Password security: A case history. Communications of the ACM, 22(11), pp.594-597.
15. NIST. (2017). Digital Identity Guidelines. NIST Special Publication 800-63B.
16. Petrov, P., Sulov, V., Parusheva, S., Penchev, B., & Collins, J. (2017). Web Technologies Used in the Home Pages of the Irish Banks. In 4th International Multidisciplinary Scientific Conference on Social Sciences & Arts SGEM 2017, pp.1135-1142. DOI: <https://doi.org/10.5593/sgemsocial2017/15/S05.142>
17. Petrov, P., Dimitrov, G., & Ivanov, S. (2018). A Comparative Study on Web Security Technologies Used in Irish and Finnish Banks. In 18th International Multidisciplinary Scientific GeoConferences SGEM 2018, Vol.18, Iss. 2.1, pp.3-10. DOI: <https://doi.org/10.5593/sgem2018/2.1/S07.001>
18. Petrov, P., Dimitrov, P., Stoev, S., Dimitrov, G. P., & Bulut, F. (2020). Using the universal two factor authentication method in web applications by software emulated device. In International Multidisciplinary Scientific GeoConference: SGEM, 20(2.1), pp.403-410. DOI: <https://doi.org/10.5593/sgem2020/2.1/s07.052>
19. Petrov, P., Malkawi, R., Shichkin, A., Dimitrov, G., & Nacheva, R. (2019). Security Certificates Used in Public Web Sites of Banks in Czech Republic, Slovakia and Hungary. TEM Journal, 8(4), pp.1224-1231. DOI: <https://doi.org/10.18421/TEM84-17>
20. Saarinen, M. J. O. (2006). Security of VSH in the real world. In International Conference on Cryptology in India (Indocrypt 2006). Berlin, Heidelberg: Springer Berlin Heidelberg. pp.95-103.
21. Saltzer, J. H., & Schroeder, M. D. (1975). The protection of information in computer systems. Proceedings of the IEEE, 63(9), pp.1278-1308.
22. Stallings, W. (2017). Cryptography and Network Security: Principles and Practice. Pearson.
23. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D. A., & De Weger, B. (2009). Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In Annual International Cryptology Conference. Springer-Verlag, pp.55-69.