QuickHash-GUI
V3.3.1

Outil graphique de hachage de données gratuit multi-plateformes

Manuel utilisateur: v3.3.1

http://www.QuickHash-GUI.org

(précédement : https://sourceforge.net/projects/QuickHash)

(Traduction US -> FR : Didier Celse)

1.0 Préambule

Ce manuel est conçu comme une aide à l'utilisateur uniquement. Ce n'est pas une autorité en matière d'algorithmes de hachage, systèmes de fichiers ou quoi que ce soit d'autre, ni en partie ni en totalité. Le logiciel est livré sans aucune garantie, y compris toute future version signée par code numérique. Utilisez-le à vos propres risques. Si vous n'êtes pas sûr des résultats, veuillez envisager de croiser vos résultats avec d'autres logiciels. Il y a de nombreux outils de hachage de données gratuits et commerciaux disponibles pour croiser vos résultats.

Les commentaires constructifs sont encouragés et bienvenus, mais les plaintes ne seront pas tolérées. Si l'utilisateur n'est pas satisfait du logiciel, il est encouragé à en utiliser un autre. Ce manuel d'utilisation, aussi complet soit-il, utilise des termes que vous ne connaissez peut-être pas, et vous ne pouvez pas toujours deviner que nous pourrions considérer comme signifiant la même chose. Par exemple. *Export* ou *Save*.

Les captures d'écran du manuel ne reflètent pas toujours entièrement la dernière version, mais les différences sont généralement mineures.

1.1 Contrat de licence

Les utilisateurs peuvent l'exécuter sur autant d'ordinateurs qu'ils le souhaitent, autant de fois qu'ils le veulent, et aussi longtemps qu'ils souhaitent. Il n'y a pas de dongles, pas de DLL, pas d'assistants d'installation ou de fichiers de licence - il suffit de cliquer et c'est parti. Tout ce qui est demandé aux utilisateurs, c'est qu'ils partagent leurs sentiments et aident à apporter des idées au développeur (tedsmith@quickhash-gui.org).

Plates-formes prises en charge : Testé sur *Microsoft Windows 10, Linux Mint19.3* et *Apple Mac OSX Big Sur*. Les utilisateurs

d'Apple Mac utilisant des versions d'OSX antérieures à *Big Sur* peuvent uniquement utiliser la v3.2.0 et les versions inférieures.

L'interface graphique QuickHash est disponible sous la licence GPL2 (voir tous les détails https://quickhashgui.org/githubfeed/) comme suit :

```
QuickHash GUI est un logiciel gratuit : vous pouvez le redistribuer et/ou le modifier sous les termes de la Licence Publique Générale GNU telle que publiée par la Fondation des Logiciels Libres, soit la version 2 de la licence, soit (à votre choix) toute version ultérieure version. Ce programme est distribué dans l'espoir qu'il sera utile, mais SANS AUCUNE GARANTIE ; sans même la garantie implicite de QUALITÉ MARCHANDE ou ADAPTATION A UN USAGE PARTICULIER. Voir la licence publique générale GNU pour plus d'informations détails. < http://www.gnu.org/licenses/ >.
```

1.2 Remerciements

QuickHash est écrit et compilé à l'aide de l'IDE du projet Lazarus et du langage Freepascal en utilisant le compilateur Freepascal. Des remerciements sont donc adressés aux développeurs et contributeurs à la fois de Lazarus et de FPC, sans lesquels QuickHash n'aurait pas existé.

Une version personnalisée des bibliothèques Freepascal MD5 et SHA-1 a été utilisée pour MD5 et SHA-1 et la bibliothèque DCPCrypt a également été utilisée pour les algorithmes de hachage SHA256 et SHA512 jusqu'à v2.8.0 de QuickHash. Avec v2.8.0 et au-dessus, cependant, les bibliothèques ont toutes deux été abandonnées en faveur de HashLib4Pascal (https://github.com/Xor-el/HashLib4Pascal) par Ugochukwu Mmaduekwe Stanley (alias Xor-el). Il est plus activement maintenu, n'a pas nécessité de peaufinage 64 bits pour permettre hachage de données volumineuses, est conforme à Freepascal v3.0, il a évité l'utilisation d'un Bibliothèque MD5 et SHA-1, et comprend une gamme complète d'algorithmes de hachage qui permettront plus

facilement l'implémentation de ces algorithmes par QuickHash dans le futur. La bibliothèque est utilisée pour tous les algorithmes de hachage courants utilisés par QuickHash. Des remerciements et une appréciation importants sont exprimés à M. Stanley pour la bibliothèque open-source.

Merci également aux auteurs de xxHash et Blake pour les avoir également rendus open-source. Notez que xxHash est sous licence BSD et Haslib4Pascal est également sous licence MIT. Blake2b est sous licence CC0.

Remerciements à Joachim Metz pour la librairie C libewf v2 (v3 pas encore incorporée) qui a été ajouté à la v3.3.0 pour ajouter la possibilité d'ouvrir et de vérifier par hachage le contenu du format de témoin expert (EWF) les fichiers d'images légal, également appelés images E01, suivant le courant dominant adoption par EnCase. Une reconnaissance est donnée aux contributeurs comme documenté ici.

libgcc_s_dw2-1.dll (bibliothèque d'exécution de bas niveau GCC) est sous licence GPL (https://gcc.gnu.org/) et la DLL a été compilée avec MSYS2

libwinpthread-1.dll est sous licence MIT, BSD (https://packages.msys2.org/package/mingw-w64-x86_64-libwinpthread) et la DLL a été compilée à l'aide de MSYS2 par Ted Smith

zLib est Copyright (C) 1995-2017 Jean-loup Gailly et Mark Adler (https://zlib.net/zlib_license.html) et la DLL a été compilée à l'aide de MSYS2

Les DLL SQLite ont été compilées à partir de la source en avril 2021 par Ted Smith à l'aide de MSYS2. Pour plus de détails concernant l'utilisation de la licence pour cela, veuillez voir ici : https://www.sqlite.org/copyright.html

1.3 Équipe de développement

Développeur principal : Ted Smith

Contact : tedsmith@quickhash-gui.org - voir les pages d'assistance et système de tickets pour les demandes de fonctionnalités et les suggestions

Référentiel GitHub
https://github.com/tedsmith/QuickHash/releases

Développeur de soutien : darealshinji

Page GitHub https://github.com/darealshinji

2.0 Présentation

QuickHash est tout simplement un hachage de données gratuit, open source, multiplateforme, rapide et facile à utiliser outil pour MicrosoftWindows, les distributions Desktop GNU/Linux telles que Mint, Zorin OS et Ubuntu, et Apple Mac OSX.

Il ne nécessite pas d'installation et peut simplement être exécuté à partir d'une clé USB ou d'un autre périphérique amovible (bien que sur Linux et OSX, le logiciel doit avoir des autorisations exécutables attribuées et le périphérique externe nécessiterait un système de fichiers pouvant stocker des autorisations exécutables - donc EXT4 ou quelque chose et non FAT32).

À partir de la v2.8.0, les quatre algorithmes de hachage "courants" habituels sont disponibles - MD5, SHA-1, SHA256 et SHA512 mais en plus xxHash (lors de l'utilisation de la version 32 bits de QuickHash, uniquement xxHash32 sera disponible, et vice versa pour 64 bits), SHA-3 (256 bits), Blake2b (256 bits), Blake3 et CRC32 ont été ajoutés au fil du temps.

L'interface est délibérément simple et se présente sous la forme d'un système à onglets - chaque onglet pour un type différent de Les données.

Text | File | FileS | Copy | Compare Two Files | Compare Two Folders | Disks | Base64 Data

2.1 Qu'est-ce qu'un hachage ?

L'explication la plus simple est que c'est comme une empreinte digitale unique de données numériques. Il y a beaucoup d'algorithmes de hachage courants, mais QuickHash est codé pour utiliser quatre des algorithmes courants courants : MD5, SHA-1, SHA256 et SHA512 avec l'inclusion du xxHash de plus en plus populaire à partir de v2.8.0 du programme. Très fondamentalement, si vous calculez la valeur MD5 de votre nom tapé, le résultat est théoriquement unique à un sur 3,4028...E38, ce qui, plus simplement, est une probabilité de un sur 340 milliard de milliard de milliard de milliard (1 undécillion), ce qui signifie que les chances que toute autre donnée numérique autres que cette chaîne de caractères générant le même hachage sont infiniment improbables (collisions collatérale).

La recherche entourant les collisions de hachage MD5 et SHA-1 est dûment documentée et le lecteur peut trouver plus d'informations à ce sujet dans des publications et des articles sur Internet ou des articles universitaires, s'il s'agit d'une préoccupation dans son domaine de travail.

XxHash, de Yann Collet https://github.com/Cyan4973 est « un algorithme de hachage extrêmement rapide, fonctionnant aux limites de vitesse de la RAM. Il complète avec succès la suite de tests SMHasher qui évalue qualités de collision, de dispersion et de caractère aléatoire des fonctions de hachage. Le code est hautement portable et les hachages sont identiques sur toutes les plateformes (little/big endian) ».

QuickHash adopte les constructions Merkle-Damgård qui permettent théoriquement une méthode de construction de fonctions de hachage cryptographiques résistantes aux collisions. Plus d'informations peuvent être lu à ce sujet en ligne.

2.2 Multiplateforme

QuickHash a été conçu à l'origine pour Linux pour permettre aux utilisateurs Linux moins avancés de facilement "et rapidement" (d'où le nom) de générer une liste de valeurs de hachage pour les fichiers à l'aide d'une interface graphique simple sans avoir à recourir à des outils de ligne de commande comme SHA1SUM. Il a été spécialement conçu pour fonctionner avec des CD de démarrage en direct comme DEFT, CAINE, PALLADIN, HELIX, PARROT SECURITY et autres. Cependant, au fil du temps, la fonctionnalité de hachage s'est améliorée et est même devenue plus rapide pour générer des valeurs de hachage par rapport à de nombreux autres outils - gratuits et commerciaux. Ce n'est donc plus seulement un outil qui permet la sélection rapide des fichiers à hacher, mais c'est aussi un outil qui calcule le hash rapidement. Donc, le nom "QuickHash" est vraiment plutôt approprié.

Il est souvent pré-intégré dans les CD DEFT, CAINE et Parrot Security Linux, donc en utilisant ces systèmes, vous aurez la puissance de QuickHash intégrée à votre instance de CD en direct sans avoir à l'utiliser séparément. Cependant, notez que les cycles de publication de QuickHash sont généralement tous les deux mois alors que les distributions amorçables ont généralement une période de rafraîchissement de cycle plus longue. Il y a aussi le package DEB construits par darealshinji qui accompagnent les binaires. Tous sont généralement disponibles sur le site Web.

En plus de la version Linux, en raison de la demande des utilisateurs de Windows, une version compatible Microsoft Windows a été faite et arbore certaines fonctionnalités qui sont nécessaires dans un environnement Windows mais pas

dans un environnement Linux. Par exemple, il est possible de calculer le hachage d'un disque physique utilisant QuickHash sous Linux en tant que root, en cliquant sur l'onglet *File* et en naviguant vers /dev/sdX ou /dev/sdXX. Ou, à partir de la v2.7.0, vous pouvez utiliser l'onglet *Disks* comme les utilisateurs de Windows. Apple mac OSX ne peut actuellement pas utiliser l'onglet *Disks*, mais peut hacher des disques en utilisant l'onglet *File* de la même manière qu'avec Linux. Dans les deux cas, le programme doit être exécuté avec les privilèges sudo.

La version Apple Mac a été développée pour la première fois avec la version 2.5.3 en 2014. Elle fonctionne dans le même esprit à la version Linux. La version Mac a d'abord été compilée sur le système d'exploitation Yosemite et version 3.0.0 avec Sierra. Depuis 2014, les trois systèmes d'exploitation sont pris en charge autant que possible, même si OSX continue d'être le plus difficile. À partir de la v3.3.0, et avec la sortie de *OSX Big Sur*, il n'était plus possible de « référencer une bibliothèque » par nom de fichier. Ainsi le lien dynamique en cache est maintenant utilisé (à partir d'avril 2021) par QuickHash-GUI, ce qui signifie que la v3.3.0 ne fonctionnera pas complètement sur toute version précédente d'OSX autre que *Big Sur*. En tant que tel, tous les utilisateurs d'OSX qui n'ont pas *Big Sur* sont limités à la v3.2.0.

2.3 Implémentation SQLite

La période d'octobre 2017 à janvier 2018 a vu la plus grande réécriture de QuickHash depuis la création du programme. Une grande partie du travail consistait à migrer la capacité existante pour interagir avec SQLite, mais les avantages en valent la peine. SQLite conserve les données très efficacement et offre aux développeurs beaucoup plus d'options pour leurs programmes. Bon nombre de ces fonctionnalités nouvelles et améliorées sont désormais disponibles en cliquant avec le bouton droit de la souris les grilles d'affichage, permettant de supprimer certaines de ces options précédentes de l'interface, pour libérer espace.

De plus, il existe maintenant de nombreuses options pour les grilles d'affichage. Elles varient un peu mais dans l'ensemble les options incluent le tri par nom de fichier, chemin, taille, numéro d'identification, doublons, connu comme la liste hash ou non et bien d'autres encore (ils peuvent être reformulés dans les futures versions, donc pas de détails complexes ici - juste faire un clic droit sur une grille d'affichage et voyez ce qui s'affiche).

Pour Windows, un fichier DLL SQLite précompilé 32 et 64 bits est généralement livré avec QuickHash. Le bon sera chargé automatiquement car QuickHash détermine de quelle architecture il s'agit en cours d'exécution. Il est important que ces fichiers ne soient pas renommés ou déplacés de l'endroit où ils sont stockés dans la structure du fichier zip. Pour les versions 3.2.0 et antérieures, cela se trouvait généralement dans le dossier racine. Depuis la v3.3.0, ils sont situés dans un sous-dossier libs du dossier du projet et ne doivent pas être déplacés ou renommés.

Si vous exécutez la version 32 bits de Quickhash sur une version 64 bits de Windows, il utilisera la version 32 bits SQLite DLL mais cela fonctionnera toujours parfaitement bien. Vous ne pouvez pas exécuter la version 64 bits sur un 32 bits Système Windows cependant.

Sous Linux, QuickHash recherchera dans plusieurs emplacements communs le fichier SQLite SO. S'il le trouve, il créera une base de données SQLite à utiliser pour cette session, et elle est supprimée lorsque QuickHash est fermé. Il s'agit de s'assurer que Quickhash ne laisse pas de restes de son activité et de s'assurer qu'il commence par une table rase à chaque exécution. L'emplacement de la base de données sera décidé par les paramètres de sécurité de votre système d'exploitation, mais il y est généralement et y résidera tant que QuickHash-GUI est ouvert :

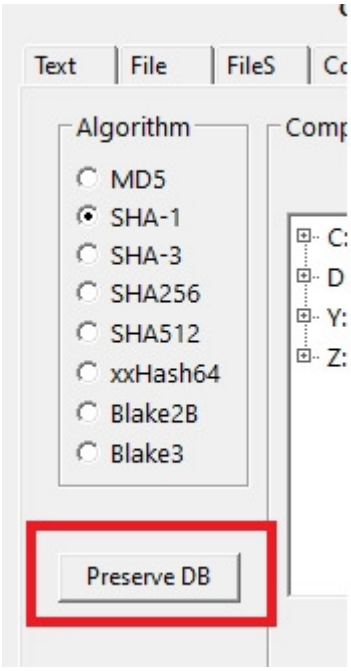
```
/home/Nom d'utilisateur/.config/QuickHash
```

Sur OSX, depuis la sortie de *Big Sur* en 2020, les bibliothèques dynamiques ne sont plus disponibles sur le système de fichiers. Au lieu de cela, le cache de l'éditeur de liens est utilisé. Cela a nécessité quelques retouches dans la v3.3.0 de QuickHash et devrait maintenant fonctionner avec un peu de chance sur *Big Sur*, mais la v3.3.0 et les versions plus récentes ne fonctionneront pas sur les anciennes versions de OS X.

Si Quickhash ne trouve pas SQLite, les onglets *FileS* et *Copy* ne fonctionneront pas comme prévu. Les autres onglets fonctionneront normalement, donc selon vos besoins, cela ne posera peut-être pas de problème.

La base de données est nommée en fonction de la date et de l'heure d'exécution de Quickhash pour permettre plusieurs instances du programme à s’exécuter (une base de données distincte sera créée pour chaque instance permettant plusieurs exécutions de Quickhash sur un ordinateur) à partir du même emplacement de lancement.

Depuis la v3.3.0, il est possible de faire une copie de la base de données SQLite facilement en appuyant sur le bouton *Preserve DB*. Cela tentera de faire une copie du fichier SQLite pour vous et de l'enregistrer dans un endroit de votre choix. Cela peut être utile si, pour une raison quelconque, QuickHash lui-même s'avère insuffisant pour votre besoins et vous pouvez utiliser à la place un outil d'exploration SQLite dédié et vos propres commandes SQL pour interroger, rechercher, filtrer, etc. Pour les utilisateurs qui traitent d'énormes volumes de données, comme plusieurs millions de fichiers, cela peut être préférable. Notez qu'il fera une copie du fichier de base de données ouvert, il est donc possible que certaines transactions peuvent ne pas avoir encore été validées, bien que cela soit peu probable. Néanmoins, il vient avec cet avertissement sécuritaire.



3.0 Interface

Onglets expliqués - Résumé rapide

Text : Pour hacher des morceaux de texte comme des paragraphes d'un fichier, un nom, une chaîne de caractères, une liste de valeurs (à hacher ligne par ligne) ou des données de clé publique qui peuvent être copiées d'ailleurs vers QuickHash

File : pour sélectionner puis hacher un fichier individuel.

FileS : Pour hacher plusieurs fichiers dans un répertoire (alias "dossier") de manière récursive.

Copy : il s'agit essentiellement d'un copier-coller, mais avec l'intégrité des données supplémentaire du hachage durant le processus. Conçu pour permettre à un utilisateur de copier des fichiers d'un endroit à un autre et d'avoir la copie processus vérifié et pris en charge par des valeurs de hachage et un journal conservé de la date et de l'heure d'origine les attributs.

Compare Two Files : choisissez simplement deux fichiers à deux emplacements différents et obtenez les hachages de tous deux comparés automatiquement.

Compare Two Folders : Pour comparer le contenu du fichier d'un dossier à un autre pour voir si tous les fichiers à l'intérieur de chaque correspondance en fonction du hachage et du nombre et, depuis la v3.3.0, les noms de fichiers sont également pris en compte.

Disks : hachez facilement des disques physiques entiers et des volumes logiques (à partir de la v2.4.0 et pour Windows, et Linux depuis la v2.7.0).

Base64 Data : nouveauté de la v2.8.3, elle permet à l'utilisateur de hacher un fichier Base64 encodé ET de générer un hachage de son homologue décodé sans que l'utilisateur ait à créer la version décodée. Ça aussi permet le décodage des données encodées en Base64, juste pour plus de commodité.

Plusieurs onglets contiennent un petit menu déroulant avec la valeur par défaut de *Set Delimiter*. C'est pour essayer d'aider les utilisateurs à choisir le caractère de délimitation à utiliser chaque fois que les grilles d'affichage sont cliqué-droit lorsque l'utilisateur choisit d'enregistrer au format CSV. La virgule est utilisée par défaut si *Set Delimiter* est laissé comme ça. Ou l'utilisateur peut en fait sélectionner ',' dans la liste, ce qui a le même effet. Les autres options recommandées sont soit le caractère de tabulation, ce qui facilite beaucoup l'utilisation de la sortie vers Excel et aide dans les cas où, par exemple, l'utilisateur peut avoir une collection d'e-mails copiés sur un fichier. Les e-mails prennent souvent le nom de leur ligne d'objet, et donc sur disque, ils peuvent inclure caractères comme une virgule. En tant que tel, la virgule peut alors provoquer une confusion de délimiteur, donc un onglet fonctionne mieux ici ; ça peut-être le trait d'union, bien que le même exemple puisse s'appliquer là aussi. L'espace est aussi inclus pour tous ceux qui ont envie de l'utiliser, bien qu'il ne soit pas recommandé pour des raisons tout à fait évidentes et QuickHash en avertira même l'utilisateur.

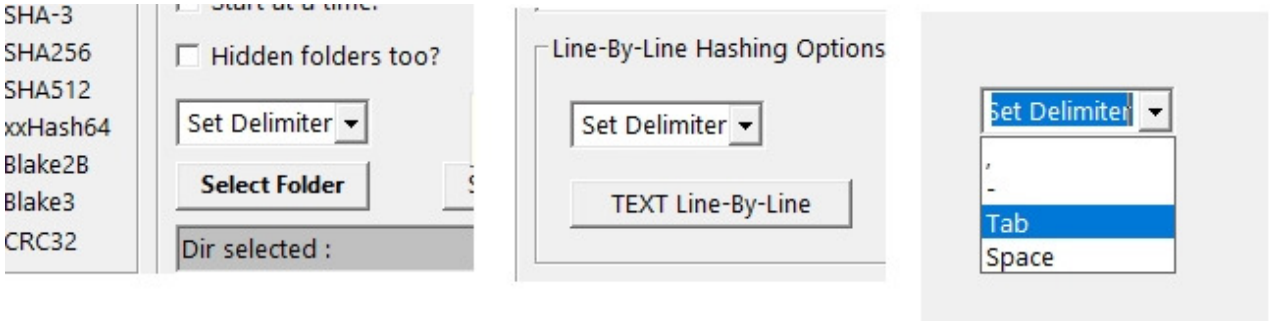


Illustration 1 : option Set Delimiter à partir de la version v3.3.0

Si l'utilisateur décide de le définir après l'analyse, alors tant que la grille d'affichage n'a pas encore été fermée, il devrait toujours prendre effet lorsque l'utilisateur déclenche un presse-papiers ou l'option Enregistrer au format CSV. Mais c'est nouveau dans v3.3.0 et peut ne pas toujours fonctionner. C'était étonnamment plus délicat à mettre en œuvre que l'utilisateur pourrait le penser.

3. Explication détaillée des onglets

3.1.1 Text :

Pour hacher des morceaux de texte comme des paragraphes ou des données clés qui peuvent être copiés à partir de quelque part à QuickHash. Vous pouvez également taper dans la zone de texte et QuickHash recalcule recalculez la valeur de hachage au fur et à mesure que vous tapez. L'utilisateur peut également basculer le hachage généré vers un hachage calculé par un autre algorithme en cliquant simplement sur l'un des autres boutons radio.

Depuis la v2.6.2, un champ *Expected Hash Value* permet à l'utilisateur de coller une valeur de hachage existante (peut-être calculé par un autre outil) et QuickHash comparera le hachage généré de l'entrée segment de texte par rapport à celui fourni par l'utilisateur. Une alerte s'affichera si les hachages ne correspondent pas. Pour annuler la comparaison, remplacez la valeur de hachage par trois points ('...').

À partir de la v2.6.5, la fonctionnalité a été ajoutée pour décomposer la zone de texte ligne par ligne. C'était à la demande d'utilisateurs car il s'est avéré que Google Adwords, et peut-être des services similaires, ont besoin de clients pour fournir leurs listes d'adresses e-mail sous forme de valeurs de hachage SHA256 en minuscules. Alors maintenant, l'utilisateur peut coller la liste de milliers d'adresses (jusqu'à 2 Go de texte) dans QuickHash et si l'utilisateur clique ensuite sur le bouton *TEXT Line-By-Line* il obtiendra un fichier de sortie séparé par des virgules contenant tout les valeurs de hachage pour chaque adresse e-mail en quelques secondes. **Sachez que les listes d'adresses e-mail en caractères majuscule généreront une sortie différente d'une liste d'adresses e-mail en minuscules ! Me@Me.com est différent de me@me.com. Faites également attention aux caractères de retour chariot qui peuvent ne pas être visibles. Utilisez Notepad++ ou un logiciel similaire pour nettoyer vos données d'entrée. Vous devez préparer votre liste en avance en utilisant Microsoft Excel ou Notepad++ et assurez-vous qu'elle est correcte.**

De plus, un deuxième bouton permet d'ouvrir un fichier texte volumineux, puis d'afficher chaque ligne de ce fichier haché ligne par ligne. Et à partir de la v2.6.7, il y a une case à cocher qui permet à l'utilisateur d'inclure ou exclure les données de texte d'origine dans le fichier de sortie. Utile pour Google Adwords où il attend juste un fichier contenant des hachages d'adresses e-mail, et non les adresses e-mail elles-mêmes qui ont été utilisées pour générer les valeurs en premier lieu. Mais il y aura d'autres occasions où l'utilisateur voudra peut-être voir le texte qui a été haché ainsi que le hachage en sortie. C'est pour ça que c'est là.

À partir de la v3.0.0, il existe des boutons pour convertir votre texte saisi en majuscules (*Make UPPER*) ou en minuscules (*Make lower*), pour plus de commodité. Les textes ASCII et Unicode devraient fonctionner correctement.

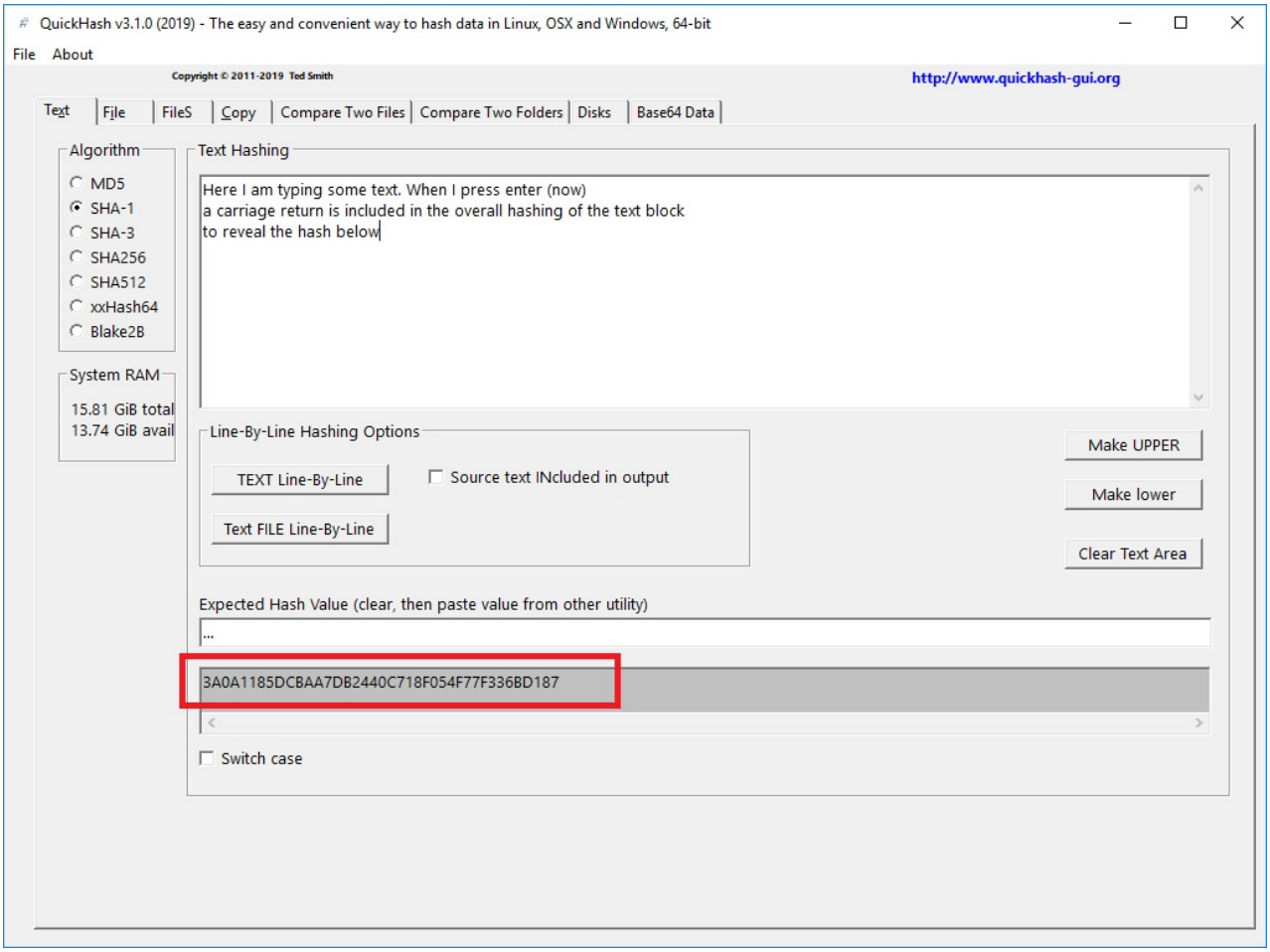


Illustration 2 : Les valeurs de hachage pour la somme du texte sont recalculées dynamiquement pendant la saisie de l'utilisateur, ou bien le bouton "Hash Line-by-Line" permet d'afficher toute la liste haché ligne par ligne

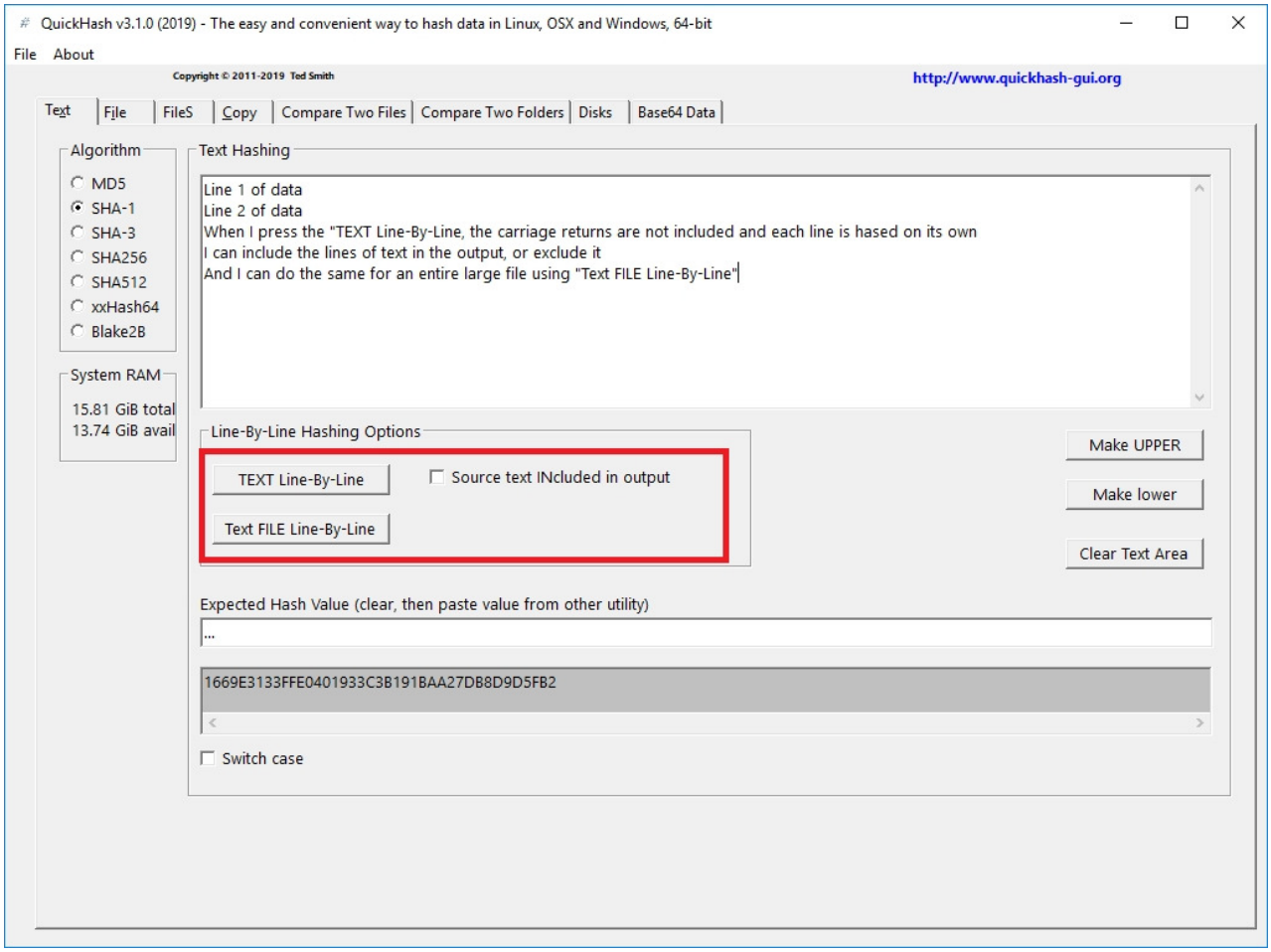


Illustration 3 : Les fonctions de hachage de texte disponibles depuis la v2.6.7

3.1.2 File : Pour sélectionner puis hacher un fichier individuel. Cliquez simplement sur *Select File* , accédez au fichier (ou faites glisser / déposez un fichier sur la fenêtre), et le hachage sera calculé. Il n'y a pas de limites de taille (depuis la v2.1 au moins) autres que celles imposées par le système de fichiers stockant le fichier sur lequel QuickHash n'a aucun contrôle, bien sûr. Il n'est donc pas nécessaire de vous inquiétez des limites de 4 Go, etc.

À partir de la v2.6.2, il y a aussi le même champ *Expected Hash Value* qui apparaît dans l'onglet *Text*, qui permet à nouveau à l'utilisateur de coller une valeur de hachage existante (peut-être calculée par un autre outil) et QuickHash comparera le hachage généré du fichier à celui fourni par l'utilisateur. La sensibilité à la casse est gérée par QuickHash, alors ne vous inquiétez pas de la conversion de votre hachage attendu de minuscules à majuscules ou de majuscules à minuscules - QuickHash s'en occupera pour vous. Une alerte sera affichée si les hachages ne

correspondent pas. À partir de la v2.8.3, vous pouvez ajouter la valeur après hachage du fichier, et si QuickHash identifie qu'il y a l'une des 9 valeurs de hachage valides dans ce champ, il verra alors s'il correspond à celui calculé. Ou bien vous pouvez y coller la valeur avant le hachage du fichier, et il vérifiera ensuite après avoir calculé le hachage s'il correspond à celui que l'utilisateur a collé.

Les caractères Unicode dans le nom de fichier ou le contenu du fichier sont également traités automatiquement.

Comme pour le texte, le hachage résultant peut être recalculé simplement en choisissant un algorithme différent dans le sélection de la boîte radio. Les fichiers plus volumineux afficheront un message indiquant *Recomputing Hash*.

La possibilité de hacher un fichier est utile, par exemple, lorsque vous avez écrit un document quelconque et que vous l'avez terminé et que vous voulez l'envoyer à quelqu'un et être certain que le fichier qu'il reçoit de vous est le même que lorsque vous l'avez terminé et envoyé. Dans ce cas, hachez le fichier avant de l'envoyer, joignez-le à votre e-mail avec une copie du hachage calculé, puis dites au destinataire d'utiliser QuickHash (ou tout autre outil de hachage d'ailleurs) pour recalculer le même hachage à l'arrivée et vérifier le calcul valeur par rapport à ce que vous mettez dans votre e-mail. Il est également très utile pour les utilisateurs qui téléchargent d'importantes données provenant d'Internet, le plus souvent des systèmes d'exploitation et des correctifs. Utilisation de QuickHash et de la valeur de hachage que le développeur Web place sur son site Web, vous pouvez être sûr que le fichier que vous avez téléchargé est le même que le fichier qu'ils y ont mis.

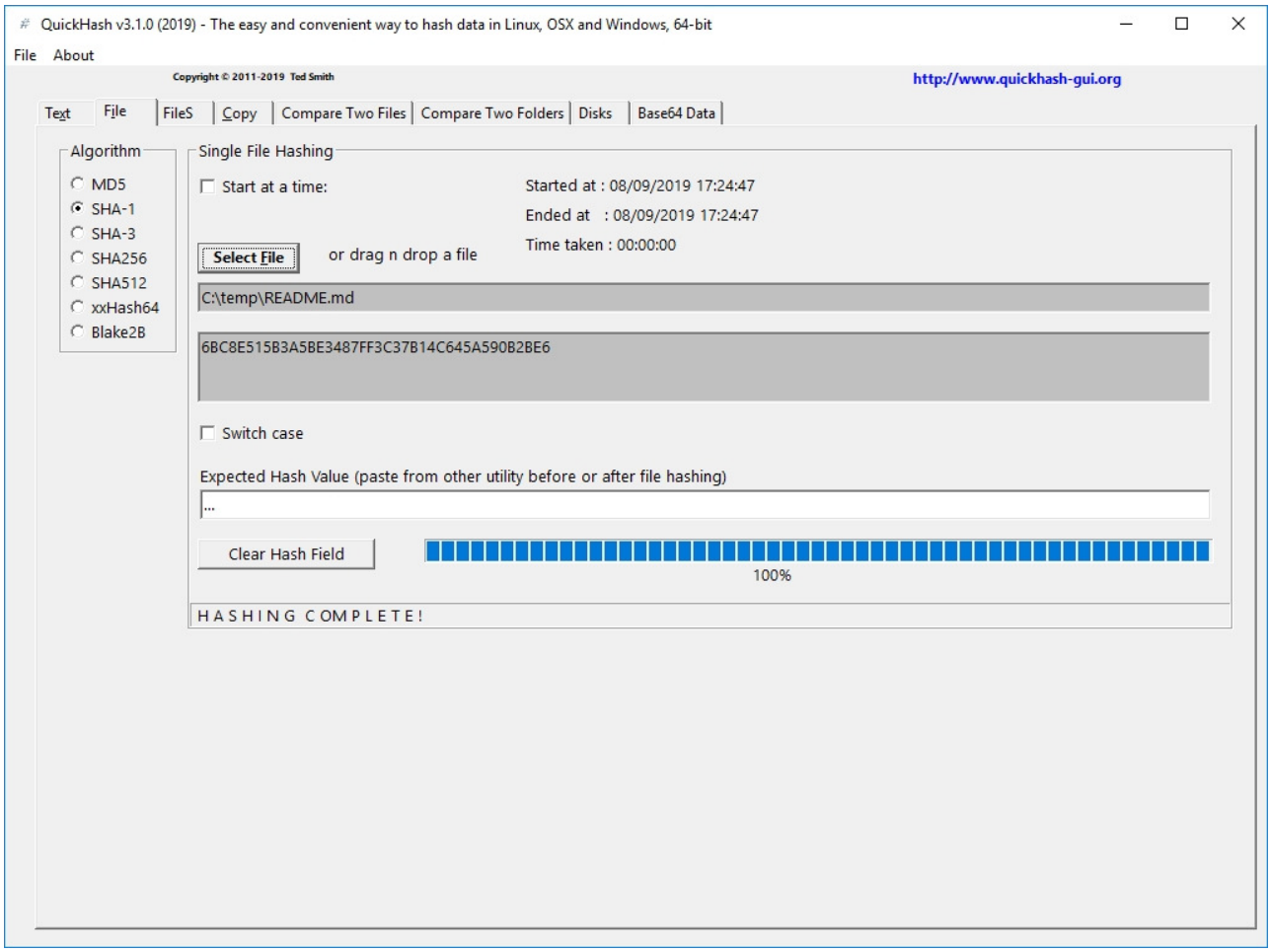


Illustration 4 : L'onglet "File" affichant un hachage calculé d'un fichier

Avec la v2.8.1. un indicateur de progression a été ajouté à cet onglet car les utilisateurs continuaient à signaler que le programme "s'était planté" lors du hachage de fichiers volumineux, alors qu'il ne pouvait tout simplement pas être interrompu. Avec v2.8.1, l'interface se mettra à jour toutes les quelques secondes dans cet onglet et donnera une idée du nombre de Mo lus en temps réel.

Sur les systèmes Linux, tout est fichier, cela peut donc inclure des disques physiques (par exemple /dev/sda) ou lecteurs logiques (/dev/sda1) si QuickHash est exécuté avec un accès root. Le hachage de disque est également disponible pour Windows et Linux en tant qu'interface graphique dans l'onglet *Disks*, mais malheureusement pas pour Apple Mac OSX - voir *Hachage de Disque,* plus loin.

En ce qui concerne les images forensiques créées par des spécialistes en criminologie numérique : il est parfois utile de hacher les morceaux individuels d'une image forensique si un logiciel forensique signale un problème avec une image, pour essayer et diagnostiquer si un morceau particulier n'a pas été déplacé ou copié correctement à partir d'une copie maîtresse. Cependant, ne confondez pas cette fonctionnalité avec la possibilité de calculer le hachage des données calculé à l'intérieur de l'image. Toutes les versions de QuickHash antérieures à la v3.3.0 ne pouvaient hacher que l'image segments, pas les données internes du fichier E01. En d'autres termes, si l'utilisateur a navigué jusqu'au premier fichier d'un ensemble d'images fragmentées (comme une image disque dur fragmentée ou une image E01 fragmentée) **le hachage résultant sera celui du morceau d'image choisi (c'est-à-dire le fichier) seulement**, pas des données acquises qui se trouvent à l'intérieur de l'image forensique entière couvrant plusieurs morceaux.

Cependant, avec la v3.3.0, pour Windows et Linux uniquement (pas OSX) la bibliothèque libewf de Joachim Metz a été ajouté. Cela signifie que si un utilisateur sélectionne un fichier .E01, QuickHash assumera alors l'utilisateur a sélectionné une image forensique qui peut inclure des segments .E02, .E03...etc. QuickHash exécute alors le libewf pour vérifier qu'il s'agit bien d'une image forensique appropriée. Si c'est le cas, QuickHash demandera à l'utilisateur s'il veut hacher les données internes de l'image, ou le segment d'image qui a été sélectionné.

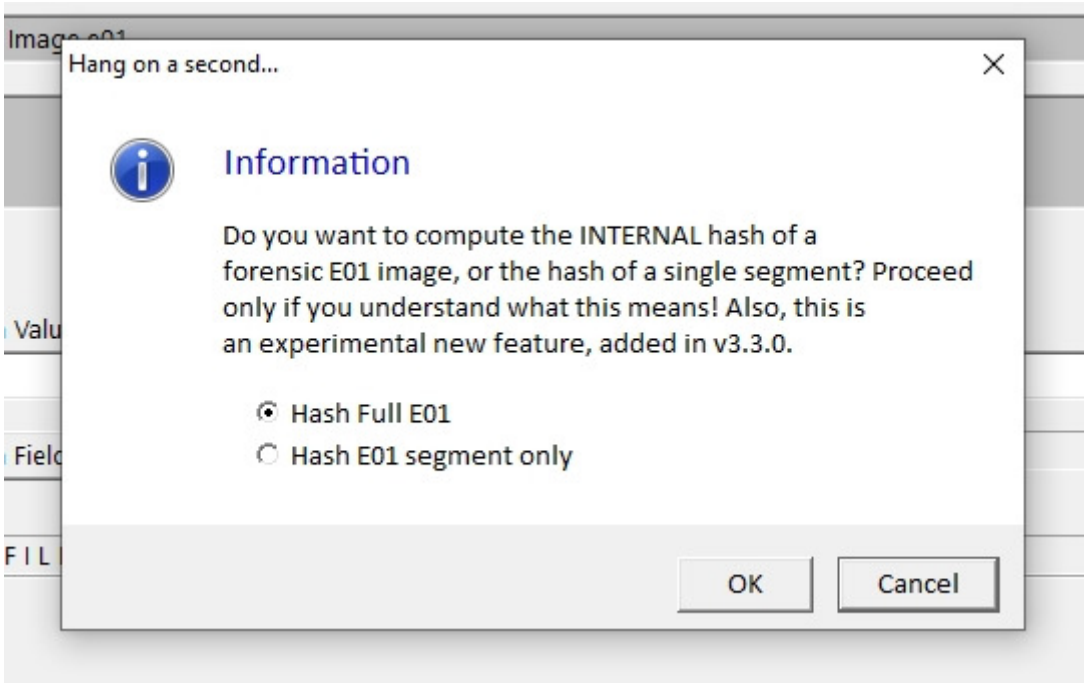


Illustration 5 : Fenêtres affichées lorsque l'utilisateur sélectionne un fichier témoin d'expertise de type E01

Si l'utilisateur choisit de hacher l'image complète, il recherchera également les valeurs de hachage MD5 ou SHA1 intégrées (celle qui est recherchée correspond à celle choisie par l'utilisateur pour le hachage), si elle est disponible, c'est celle que l'outil forensique a inséré au moment de l'acquisition. QuickHash l'affichera ensuite dans le champ *Expected Hash Value*, et vérifie que le hachage calculé correspond au hachage stocké. Ceci n'est que pour MD5 et SHA-1 actuellement. Si aucune valeur de hachage intégrée n'est trouvée, il rapportera simplement la valeur calculée hachage de l'image forensique complète comme d'habitude et l'utilisateur peut ensuite comparer cette valeur avec un autre outil, ou simplement en prendre note à des fins d'audit.

Si l'utilisateur choisit de hacher uniquement le segment E01 sélectionné, en revanche, le hachage sera calculé pour ce segment de fichier, comme pour tout autre fichier. Si l'utilisateur change d'algorithme sans changer le fichier choisi, on lui demandera à nouveau quel type de hachage doit être menée, car il s'agit d'une distinction assez importante en termes de hachage de données et non quelque chose à supposer. Les résultats des étapes de calcul et de vérification du hachage d'image E01 ont été contre-vérifiés

pendant le développement à l'aide de X-Ways Forensics v20.0 et les résultats sont les mêmes que ceux observés dans l'illustration ci-dessous (bien qu'évidemment X-Ways Forensics calcule le hachage plus rapidement):

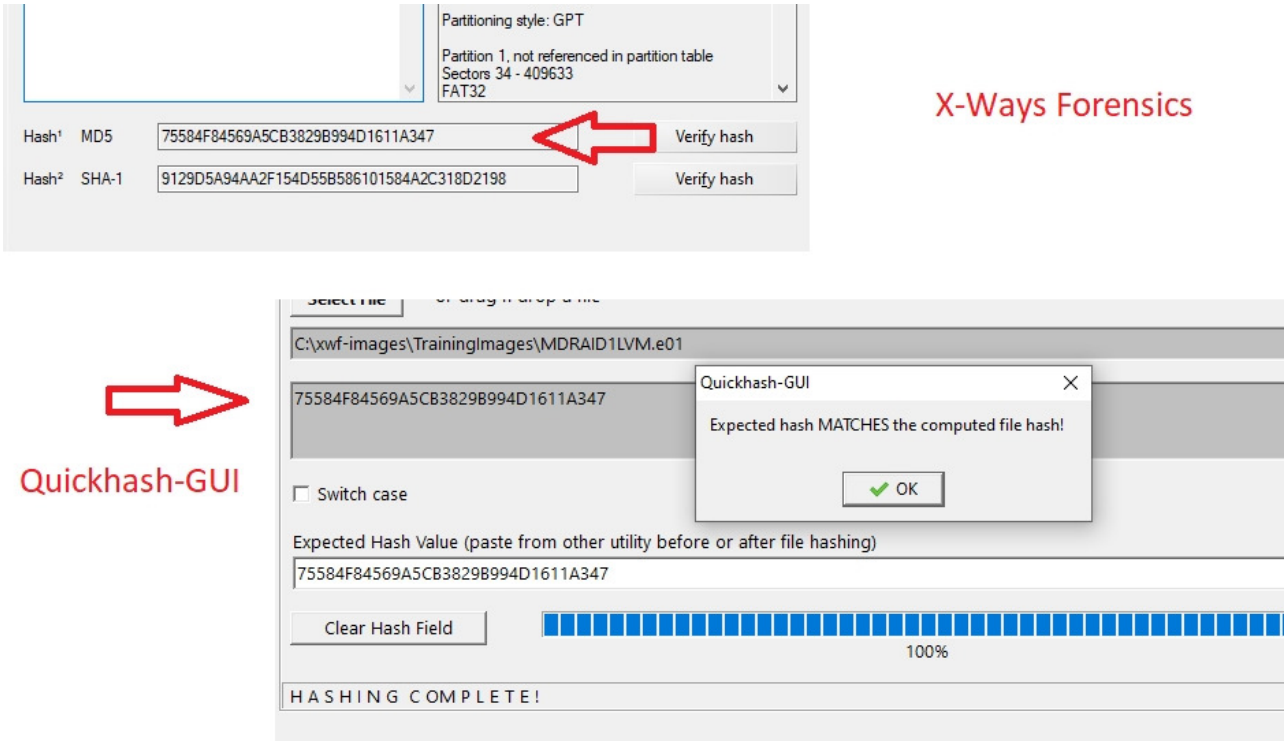


Illustration 6 : résultats de hachage E01 recoupés avec des outils forensiques

Notez que cette fonctionnalité de hachage E01 n'est effectuée qu'à l'aide de l'onglet *File*. Les fichiers E01 trouvés ailleurs lors de l'utilisation de QuickHash, comme lors de l'utilisation de l'onglet *FileS*, des onglets *Copy* ou *Compare Two Folders* ne sont pas vérifiés de cette manière car il est supposé par Quickhash que l'utilisateur n'effectuera une telle activité que sur une image forensique complète, et non sur une qui s'est retrouvée dans un dossier avec des tas d'autres fichiers que ce soit par accident ou par l'emploi d'un fichier au hasard de la stratégie de gestion.

Cette nouvelle fonctionnalité est conçue pour les experts en criminologie numérique et ne doit pas être utilisée par des personnes qui ne comprennent pas les actions qu'ils mènent. Comme toujours, la vérification à double outil avec des produits comme X-Ways Forensics, FTK Imager, NUIX ou des produits open source comme le Sleuthkit, etc. est conseillée. Et libewf peut également être installé en tant que

package sur votre système dans le cadre de la suite libewftools.

Notez que pour des raisons d'intégrité, si vous avez une image E01 forensique dans LocationA, et que vous avez vérifié à l'aide de n'importe quel outil forensique à cet emplacement actuel, puis vous le copiez à l'emplacement B, vous pouvez hacher chacun de ces segments d'image dans LocationA et LocationB comme décrit ci-dessus en utilisant l'onglet *FileS*. Si les hachages correspondent tous à chaque segment de fichier et que l'image d'origine a été vérifiée où il utilisait un outil forensique, vous pouvez être assuré que la copie de l'image dans LocationB est également valide du fait de la nature même du principe du hachage. Il n'est sans doute pas nécessaire d'ouvrir et recommencez avec un outil forensique pour recalculer le hachage interne si les hachages des segments dans LocationA correspondent à LocationB, et qu'il a été d'abord vérifié avec un outil forensique dans LocationA.

Les utilisateurs exécutant la version 32 bits de Quickhash-GUI peuvent rencontrer une erreur concernant les fichiers DLL manquants au-delà de ce qui est inclus dans le dossier libs\x86. Par exemple : vcruntime140.dll, et d'autres, font partie du NET Framework et du package redistribuable Microsoft Visual C++ 2015 (ou plus récent). La plupart des versions modernes et mises à jour de Windows 7 et 10 seront probablement déjà installées. Mais si vous voyez des avertissements concernant ces DLL manquantes, installez au moins *Microsoft Visual C++ Redistributable Package* et réessayez. Par exemple celui trouvé ici

https://www.microsoft.com/en-us/download/details.aspx?id=52685

Enfin, la bibliothèque libewf a été introduite avec la v3.3.0 pour satisfaire la demande. On pourrait appeler ça un nouvel ajout expérimental à Quickhash. Avec cette mise en garde, le développeur n'accepte ni ne suppose aucune responsabilité quant au résultat ou à votre compréhension des résultats. La criminologie numérique est une méthode informatique minutieuse, et cette nouvelle fonctionnalité est

une tentative d'implémenter cette fonctionnalité dans ce qui est un outil de hachage de données populaire multiplateforme, pour plus de commodité et d'enthousiasme. Mais finalement, c'est un projet de passe-temps. Les outils d'investigation numérique coûtent des milliers d'euros et occupent, généralement, des dizaines ou des centaines de développeurs. Quickhash est principalement développé par une seule personne. Et c'est distribué gratuitement.

Le support E01 sera-t-il ajouté pour Apple OSX ? Probablement un jour, mais pas encore. La sortie de *Big Sur* a pénalisé durement les développeurs. Des semaines ont été passées à essayer de comprendre ce qui se passait avec leur nouvelle bibliothèque système. Et compiler une nouvelle bibliothèque et la distribuer n'est toujours pas clair. Mais peut-être qu'à l'avenir, ça le sera. Là où il y a une volonté, et au moins une certaine demande, il y a un peu d'espoir.

3.1.3 FichierS :

C’est pour hacher plusieurs fichiers dans un répertoire (alias "dossier") de manière récursive. En termes simples, choisissez un répertoire et QuickHash trouvera tous les fichiers sous ce répertoire et dans ses répertoires enfants et calculer les hachages pour tous les fichiers, en affichant les résultats à l'écran. Si vous avez de nombreux fichiers, choisir *xxHash* comme algorithme de hachage ça sera considérablement plus rapide que n'importe lequel des les autres.

Il y a plusieurs options dans cet onglet :

1. Ignorer les sous-répertoires (*Ignoring sub-directories ?)*
2. Aussi les dossiers cachés ? (*Hidden folders too ?)*
3. Choisissez les types de fichiers ? (*Choose file types ?)*
4. Démarrer à telle heure : (*Start at time :*)
5. Charger la liste de hachage ? (*Load HashList*)

Les options 1 à 5 exigent toutes que l'utilisateur coche la case avant de commencer l'analyse.

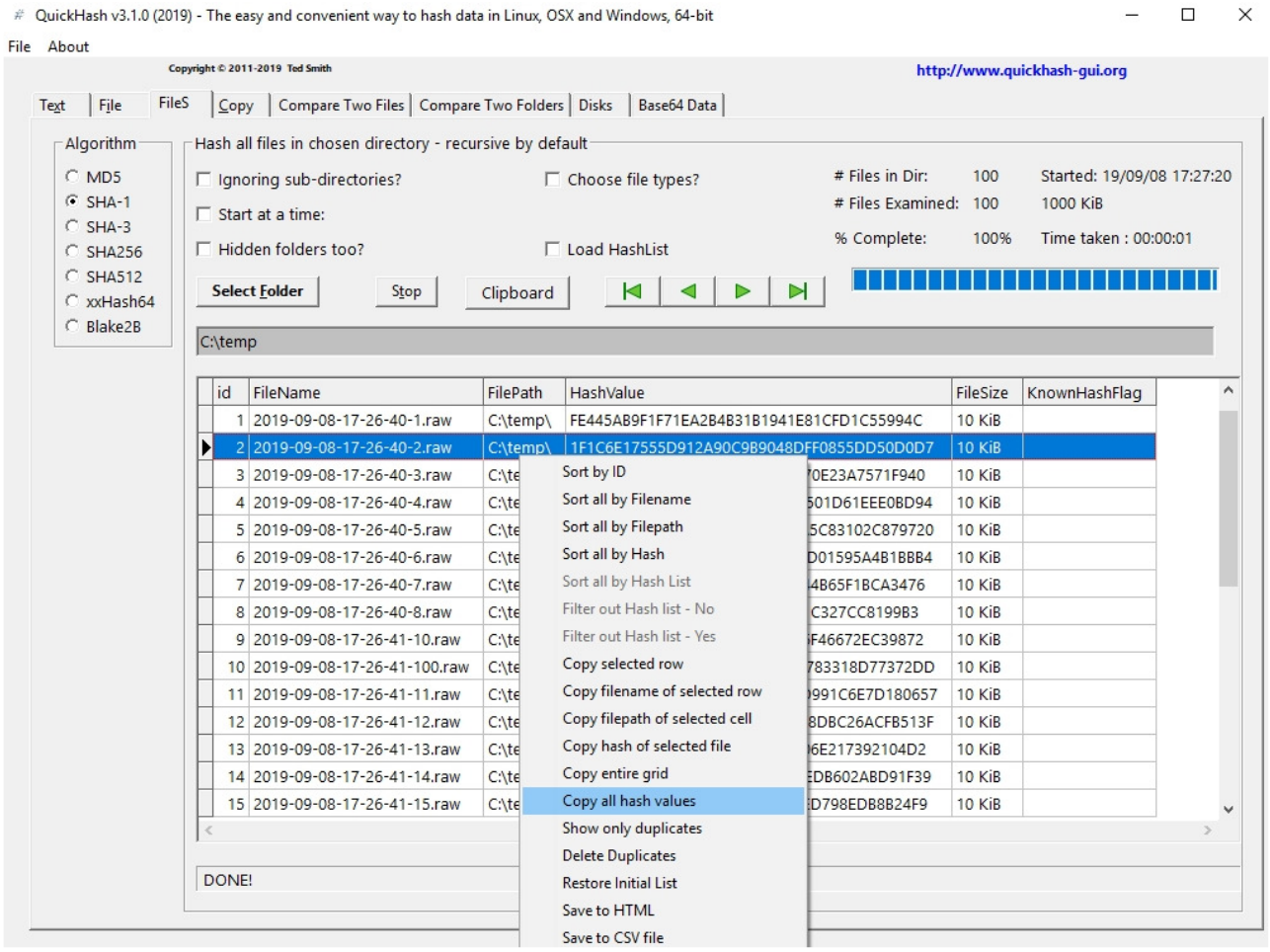


Illustration 7 : Fichiers hachés dans l'onglet FichierS

L'option *Ignoring sub-directories ?* permet à l'utilisateur de calculer des hachages de fichiers à la racine du répertoire choisi mais aucun des fichiers situés dans les sous-répertoires enfants qui peuvent se trouver en dessous ce répertoire racine choisi.

L'option *Hidden folders too ?* nécessite quelques explications détaillées. Sous Windows, les fichiers cachés seront trouvés et hachés par défaut, mais uniquement s'ils résident dans des dossiers non cachés.

Les fichiers qui résident dans des dossiers cachés, qu'ils soient eux-mêmes cachés ou non, ne seront trouvés que si cette option est cochée. En cochant la case cependant, tous les fichiers, cachés ou non masqués, qui résident dans des dossiers masqués ou non masqués, seront trouvés. Les captures d'écran ci-dessous le démontre.

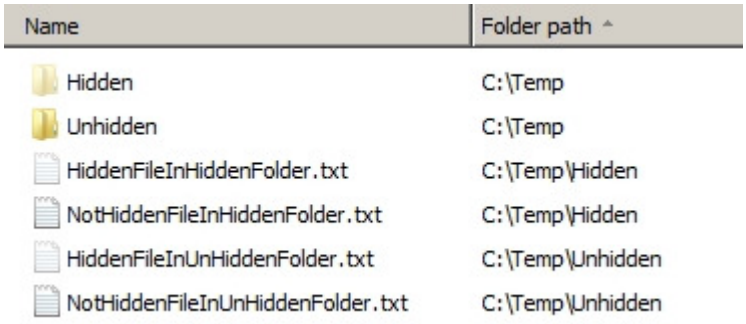


Illustration 8 : Fichiers avec différents attributs de systèmes de fichiers

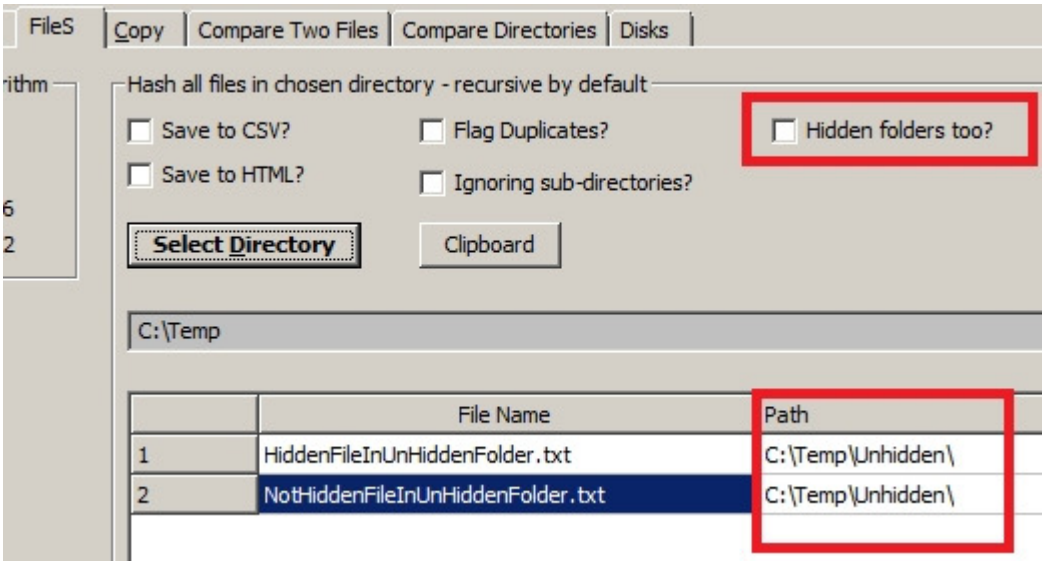


Illustration 9 : QuickHash ignorera les fichiers dans les dossiers cachés si demandé

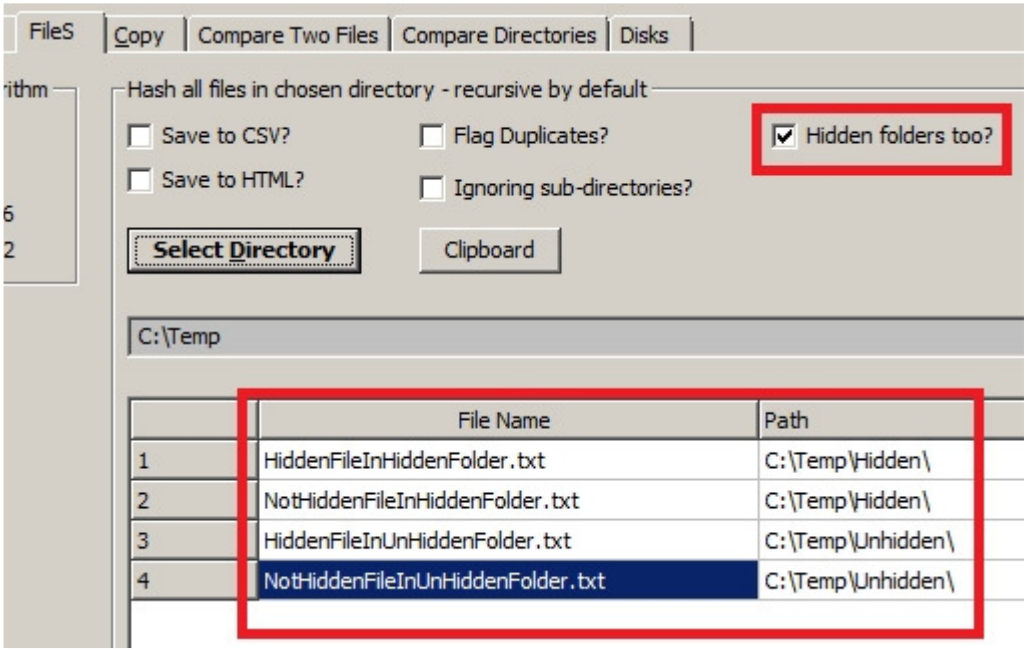


Illustration 10 : QuickHash examinera les fichiers de dossiers cachés, si demandé (capture d'écran v2.6.3)

L’option *Choose file types ?* a été ajoutée à l'onglet *FileS* dans la v2.6.4 à la demande d'utilisateurs, ce qui signifie que l'utilisateur peut hacher de manière récursive le contenu d'un dossier entier et de ses sous-dossiers, mais uniquement les fichiers qui ont l'extension ou les extensions saisies seront analysés. Chaque valeur doit être séparée par un point-virgule (';'). Notez que l'analyse de signature d'en-tête de fichier n'est pas effectuée, analyse des extensions de nom de fichier seulement.

Depuis la v2.6.4, QuickHash trouvera également les fichiers et dossiers qui dépassent la valeur MAX_PATH de 260 caractères imposés par MS Windows. Les systèmes de fichiers sous-jacents de la plupart des systèmes d'exploitation, y compris NTFS, prend en charge les chemins de fichiers bien plus longs que 260 caractères, mais Windows lui-même ne le fait pas actuellement, même si le système de fichiers sous-jacent le fait. Il est cependant possible que certains logiciels dépassent délibérément cette limite, ce qui signifie que des fichiers peuvent exister dans des chemins auxquels l'utilisateur ne peut généralement pas accéder. Avec la v2.6.4, ces fichiers seront trouvés et hachés jusqu'à une longueur de 32 000 caractères pour Windows et 4K pour Linux. Notez cependant que les noms de fichiers, en particulier, ont une limite finie qui est généralement mesuré en longueur de caractère (en gardant à l'esprit les caractères Unicode = 1 caractère, mais peut être égal à 2 octets avec UTF8, ou même 4 octets avec UTF16).

Les résultats peuvent également être copiés dans le presse-papiers à partir de la grille d'affichage en cliquant sur le bouton *Clipboard*, qui sera « cliquable » une fois l'analyse terminée, désactivé jusque-là.

L'option *Start at time :* permet à l'utilisateur de programmer une date et une heure dans le futur pour démarrer le hachage. Cependant, sachez que dans le développement, certaines incohérences ont été remarquées. Cela semble fonctionner correctement sur certaines architectures de processeur, mais pas toutes. Votre compteur peut varier

alors essayez-le avant de l'utiliser pour quelque chose d'important.

L'option *Load Hashlist ?* est nouvelle dans la v3.0.0 et était peut-être la fonctionnalité la plus fréquemment demandée de QuickHash par les utilisateurs au fil des ans. Cela permet à l'utilisateur d'importer une liste de n'importe quel nombre (limite encore à découvrir) de valeurs de hachage existantes qui peuvent avoir été générées à une date antérieure par QuickHash ou peut-être par un autre outil de hachage de données ou un outil d'investigation numérique. La liste doit être juste une colonne de valeurs de hachage sans ligne d'en-tête. Une fois sélectionné, QuickHash traitera rapidement les valeurs. Lorsque l'utilisateur sélectionne ensuite un dossier de fichiers, il calcule les hachages dans le dossier, puis rechercher si le hash correspondant existe dans la liste importée par l'utilisateur (si la case à cocher est cochée). Si c'est le cas, il ajoutera *Yes* dans la dernière colonne la plus à droite de la grille d'affichage. A l'inverse, s'il est introuvable, QuickHash affichera *No*. À la fin, trois options seront activées avec le menu clic-droit pour permettre à l'utilisateur de trier ou de filtrer les valeurs si nécessaire. La colonne sera vide si aucune liste de hachage n'a été importée.

A noter qu'un bug a été identifié en mai 2018 et corrigé en septembre 2018 avec la v3.0.3 où les listes de hachage en minuscules n'étaient pas converties en majuscules, ce qui signifie que les valeurs identiques n'étaient pas identifiées comme tel. Cela a été corrigé dans la v3.0.3 afin que toutes les listes importées soient converties en majuscules si nécessaire.

La liste restera en mémoire jusqu'à ce que QuickHash soit fermé. La liste peut cependant être complétée après un traitement si une autre liste doit être ajoutée. Cliquez à nouveau sur le bouton pour choisir un deuxième fichier de valeurs de hachage à importer et elles seront ajoutées à la première.

Une fois QuickHash fermé, la liste sera également effacée de la mémoire. Après avoir relancé QuickHash l'utilisateur devra réimporter toutes les listes de hachage si elles sont nécessaires, mais cela ne devrait pas prendre longtemps. Les versions futures peuvent prendre en charge la conservation de ces valeurs à long terme, on vera.

Notez que l'importation de la liste de hachage était nouvelle dans QuickHash v3.0.0 et ce n'était pas une quantité insignifiante de travail à mettre en œuvre, il peut donc encore y avoir quelques améliorations dans les futures versions.

Les hachages calculés de l'onglet *FileS* ne peuvent pas être recalculés dynamiquement, contrairement aux deux premiers onglets. Le glisser-déposer de répertoires n'est pas non plus possible dans cet onglet.

Depuis la v3.1.0, la colonne de hachages calculée pour les fichiers peut être exportée seule, sans toutes les autres valeurs comme le nom de fichier, etc. Ceci est utile pour la création de vos propres listes de hachage et peut inclure ou exclure une ligne d'en-tête.

Les chemins réseau UNC peuvent également être sélectionnés de cette manière si la boîte de dialogue de votre système d'exploitation le prend en charge.

La nouveauté de la v3.3.0 est la possibilité de calculer les sommes de contrôle CRC32. Un vieil algorithme certes, mais toujours d'actualité, en usage courant aujourd'hui, semble-t-il, et ajouté à la demande générale de nombreux magnats des médias qui utilisent QuickHash pour les fichiers vidéo et audio. Si l'utilisateur sélectionne CRC32 dans l'onglet *FileS*, l'utilisateur être maintenant invité à effectuer une action basée sur la fenêtre de dialogue suivante :

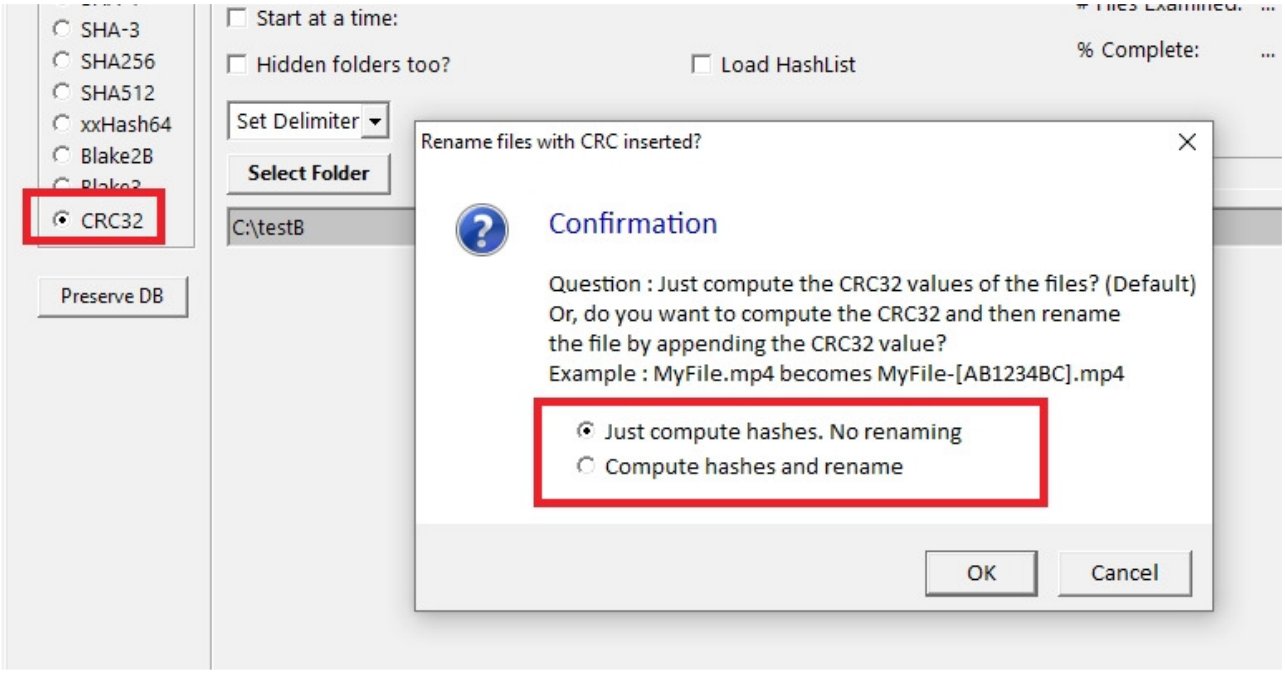


Illustration 11 : Fenêtre de dialogue affichée dans l'onglet FileS lorsque CRC32 est sélectionné

Le comportement par défaut est que les fichiers seront simplement hachés et répertoriés tels qu'ils sont dans cet onglet comme pour tout autre algorithme. Mais, utilement, la deuxième option consiste à demander à QuickHash de renommer (ajouter) le nom de fichier avec le hachage CRC32 nouvellement calculé du fichier. Ainsi, par exemple, *MyFile.mp4* devient *MyFile-[AB123456AA].mp4* et ces noms sont ensuite répertoriés dans la grille d'affichage et peuvent être enregistrés à partir de là ou copié dans le presse-papiers si nécessaire.

Attention ici. Les fichiers de l'utilisateur seront renommés s'il choisit *Compute hashes and rename.* Assurez-vous d'abord d'en avoir une sauvegarde et si vous n'en avez pas, et si le résultat n'est pas celui que vous attendiez, ne vous plaignez pas. La sécurité des données, la réalisation de sauvegardes et le test de la capacité de restauration sont la responsabilité des détenteurs de données eux-mêmes, et non de QuickHash ou de quiconque le développant. De telles plaintes vont aboutir dans ce paragraphe spécifique de ce manuel d'utilisation. Et les habituelles mises en garde concernant l'utilisation de logiciels libres open source (aucune garantie, etc.) seront réitérées.

Si l'utilisateur clique sur *Cancel*, la tentative de hachage est complètement abandonnée et l'utilisateur devra re-sélectionner le dossier des fichiers qu'ils souhaitent hacher comme il l'a toujours fait pour déclencher la prochaine tentative de hachage.

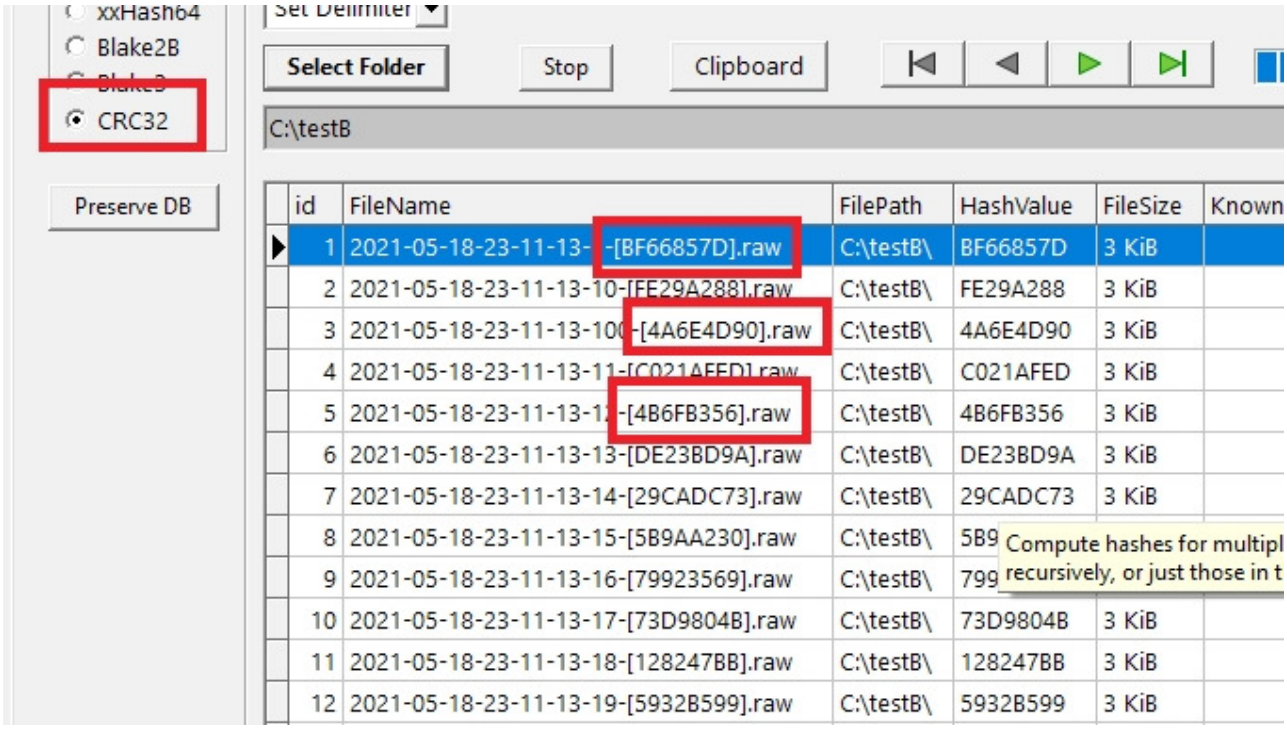


Illustration 12 : Fichiers renommés lors du calcul de la somme de contrôle CRC

3.1.4 Copy :

L'onglet Copy permet à l'utilisateur de sélectionner un dossier contenant les fichiers qu'il souhaite copier et un autre dossier vers lequel il souhaite les copier. Les dossiers source et destination peuvent être un dossier local ou un lecteur réseau mappé ou une adresse réseau UNC non mappée. Des dossiers de sources multiples peuvent être sélectionnés comme source (en utilisant Ctrl et le clic gauche de la souris) mais un seul dossier peut être sélectionnée comme destination (les dossiers d'origine seront reconstruits dans le dossier de destination).

Après avoir sélectionné les dossiers source et destination, en cliquant sur *Go*, QuickHash hachera les fichiers dans le dossier source, puis les copiera les dans le dossier de destination (tout en reconstruisant le chemin du dossier des fichiers d'origine, par défaut) où il re-hache ensuite les fichiers pour vérifiez qu'ils correspondent aux valeurs de hachage des calculs d'origine. Il s'agit d'une copie forensique. Comme pour l'onglet *FileS* il existe des options pour enregistrer les résultats au format CSV ou HTML et les sous-répertoires du répertoire source peuvent être ignorés. Notez également que l'utilisateur a la possibilité de ne pas reconstruire la structure du répertoire source dans la destination en cochant l'option *Don't rebuild path?.*

Il y a 8 options uniques à cet onglet qui nécessitent une explication.

1. Seulement répertorier les répertoires ? (*Just LIST Directories*)

2. Seulement répertorier les sous-répertoires et fichiers ? (*Just LIST sub-directories and files ?*)

3. Enregistrer les résultats (CSV) ? (*Save Results CSV*)

4. Commencer quand ? (*Start at a time ?*)

5. Ignorer les sous-répertoires ? (*Ignore sub-directories*)

6. Choisissez les types de fichiers ? (*Choose file types ?*)

7. Ne pas reconstruire le chemin ? (*Don't rebuild path ?*)

8. Copiez les fichiers cachés ? (*Copy hidden files ?*)

Les options 1 et 2 servent simplement à lister (donc 'LIST' est en majuscule) soit la structure du dossier source choisi (à l'exclusion des fichiers) ou pour répertorier les noms de dossiers ET les noms de fichiers du dossier source choisi, mais sans réellement hacher aucun des fichiers qu'il contient. Ceci est utile lorsqu'un l'utilisateur a besoin d'une

représentation textuelle d'une structure de répertoires à coller dans un autre logiciel ou un rapport.

L'option *Choose file types ?* permet à l'utilisateur de spécifier les types de fichiers à rechercher, hacher et copier. Ainsi, par exemple, si l'utilisateur n'est intéressé que par les fichiers *.doc*, en entrant simplement *".doc;"*, seul ces types de fichiers seront trouvés et copiés. Plusieurs extensions peuvent être utilisées si elles sont séparées par un ';' UNIQUEMENT (pas d'espaces). Notez que cette identification de type est effectuée par nom de fichier uniquement – mais pas l'analyse précise de la signature de l'en-tête de fichier (qui n'est actuellement pas disponible dans QuickHash).

L'option *Don't rebuild path ?* permet à tous les fichiers trouvés dans le dossier source et ses sous-dossiers d'être simplement copiés à la racine du dossier de destination sans reconstruire le chemin d'origine dans le chemin de destination. Évidemment, bien que deux fichiers du même nom ne puissent pas exister dans le même répertoire sur le même système de fichiers alors que deux fichiers du même nom peuvent exister dans un dossier et n'importe lequel de ses autres dossiers. Pour tenir compte de cela, lorsque cette option est activée (elle est désactivée par défaut) QuickHash vérifiera l'existence d'un fichier portant le même nom dans le répertoire de destination pour chaque fichier qu'il copie. Lorsqu'il est trouvé, il renommera la deuxième, troisième, quatrième (et ainsi de suite) instance du fichier en le renommant *FileName.ext_DuplicateNameX* où X est le compteur de noms de fichiers détectés en double. Notez qu'il ne s'agit pas d'une vérification du hachage de fichier basé sur le contenu - simplement nom de fichier et est fourni en raison des restrictions du système de fichiers.

L'option *Copy hidden files ?* est désactivée dans la version Windows, car les fichiers cachés dans les deux les dossiers cachés et non cachés sont trouvés par défaut avec cet onglet particulier dans QuickHash (c'est différent de l'onglet *FileS* et de son option *Hidden folders too ?*). Cependant, sous Linux et Apple Mac, la case est activée, en raison de la façon dont les fichiers et les dossiers sont essentiellement des "fichiers"

sur ces systèmes, un dossier caché doit donc être traité différemment d'un fichier caché. Si vous voulez de tels fichiers en utilisant l'un de ces systèmes, cochez cette case. Mais les utilisateurs de Windows n'ont pas à s'en préoccuper.

Les hachages ne peuvent pas être recalculés dynamiquement dans cet onglet, contrairement aux deux premiers onglets.

Le glisser-déposer de dossiers n'est pas non plus possible dans cet onglet.

Sous Windows, à partir de la v2.6.4, les fichiers qui résident dans un dossier dont la longueur dépasse 260 caractères peuvent maintenant être trouvés par QuickHash et copiés. Les versions précédentes du programme ne pouvaient copier les fichiers que vers un dossier de plus de 260, mais il n'a pas pu les lire\les détecter. À partir de la v2.6.4, il devrait être capable de faire les deux.

Cette difficulté courante est due à une restriction appelée 'MAX_PATH' (voir la description précédente) et est une limitation MS Windows. Ce n'est pas une limitation du système de fichiers NTFS qui peut permettre jusqu'à 32K caractères. Ce n'est pas non plus une limitation Linux, qui autorise jusqu'à 4K caractères.

Notez cependant que si les fichiers trouvés sont déjà proches de la limite de 260 caractères, mais pas au-delà, il est probable qu'une fois copié, la longueur du chemin dépasse 260 caractères. QuickHash va gérez cela en implémentant un contournement du système de fichiers.

3.1.5 Comparer deux fichiers (*Compare Two Files*) :

Il arrive souvent qu'un fichier existe à deux endroits différents, par exemple une sauvegarde de fichier. Cet onglet permet à un utilisateur de choisir spécifiquement un fichier, puis de le hacher contre un autre fichier automatiquement, peut-être en les laissant toute la nuit s'ils sont volumineux.

Cela évite le nécessité pour l'utilisateur d'avoir à hacher tous les fichiers dans le dossier de ces deux fichiers respectifs (en utilisant l'onglet *FileS*), ou sans avoir besoin de hacher FileA d'abord, puis de choisir manuellement FileB ensuite à l'aide de l'onglet *File*.

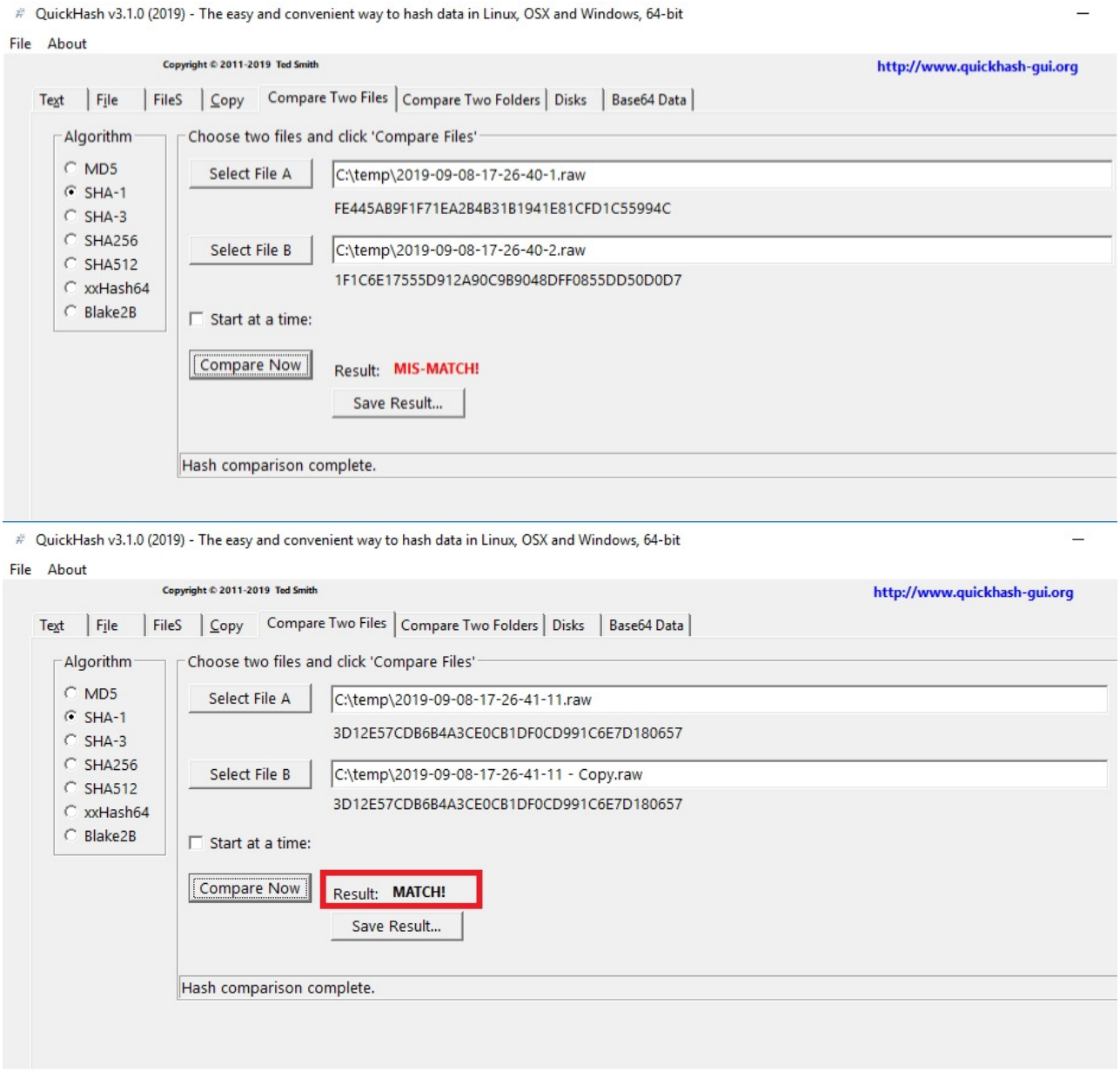


Illustration 13 : Comparaison des hachages de deux fichiers identiques et non identiques

Les résultats peuvent être enregistrés dans un fichier texte, si nécessaire, une fois le hachage terminé et en cliquant sur le bouton *Save As*.

À partir de la v2.8.1, si **l'utilisateur clique une fois** avec la souris sur la valeur de hachage générée, la valeur de hachage et le l'algorithme choisi sera copié dans le presse-

papiers. Cela s'applique uniquement à l'onglet *Compare Two Files*, pour l'instant.

Bien sûr, l'utilisateur peut simplement prendre une capture d'écran des résultats !

3.1.6 Comparer deux dossiers (*Compare Two Folders*) :

Cela permet à l'utilisateur de comparer le contenu du fichier de deux dossiers. L'utilisateur doit sélectionner un dossier comme source (FolderA) puis un second dossier pour comparer la source (FolderB). QuickHash comptera et hachera alors tous les fichiers du dossier A et du dossier B, en stockant les valeurs dans une liste en mémoire. Une fois terminé, il compte ensuite le nombre de fichiers dans chaque dossier, puis il les hache tous. Par défaut, les lecteurs mappés sont supposés et affichés en arborescence. Mais si l'utilisateur coche "Mode UNC", il est possible de coller deux chemins réseau UNC à la place. Cependant, l'utilisateur ne peut pas comparer un lecteur local sur un lecteur mappé et un chemin réseau UNC. Si cette exigence existe, veuillez monter temporairement le chemin réseau en tant que lettre de lecteur, puis utiliser le mode non-UNC à la place, comme un routage par défaut. Ensuite, démontez simplement le chemin réseau après une analyse. *Remarque : si le programme est exécuté avec des droits d'administrateur, les lecteurs réseau mappés peuvent ne pas apparaître pas dans l'arborescence. Si cela se produit, quittez le programme et lancez-le en tant qu'utilisateur normal non administrateur.*

La liste des hachages est également conservée en mémoire où ils sont ensuite comparés très rapidement.

Cependant, le système de comparaison nécessite une certaine compréhension et ne doit pas toujours être pris comme valeur nominale. De manière générale, si les deux listes de hachage correspondent et que le nombre de fichiers correspond, il signalera une correspondance même si les noms de fichiers sont différents. La fonctionnalité est conçue pour comparer deux dossiers qui sont supposés identiques

et non différents. Mais une foule de nouvelles fonctionnalités ont été ajouté à la v3.3.0 pour aider à identifier les variances là où elles existent (voir ci-dessous).

Notez que les noms de fichiers ne sont pas pris en compte dans cette analyse pour une correspondance de hachage ; c'est le contenu de chaque fichier qui est analysé. Mais, la grille d'affichage peut dérouter les utilisateurs car si un fichier est trouvé dans le dossier A appelé "Bob.doc" et le même fichier exact (par hachage) est dans le dossier B appelé "Alice.doc", une correspondance sera signalé (car le contenu binaire est le même) mais la grille d'affichage affichera un champ vide pour chaque dossier, pour chacun des deux fichiers, ce qui peut faire penser à l'utilisateur qu'il y a une non concordance. Ce n'est pas le cas. Il indique simplement qu'un fichier appelé Bob.doc a été trouvé à un emplacement avec un hachage, et le même hachage a également été trouvé dans l'autre dossier appelé Alice.doc, mais un fichier appelé Bob.doc avec le même hachage n'a pas été trouvé. Ceci est conçu pour être utile et pour mettre en évidence la disparité tout en en veillant à ce que les valeurs de hachage de chaque fichier soient toujours répertoriées. En fin de compte, la sortie est souvent mieux analysée à l'aide d'outils d'analyse de données comme Excel.

Si vous avez 3 fichiers appelés A.doc, B.doc et C.doc dans le dossier A et 3 fichiers appelés D.doc, E.doc et F.doc dans le dossier B ayant **exactement le même contenu**, cela sera signalé comme une correspondance, car le "contenu hachable" et le nombre de fichiers sont les mêmes, même si les noms de fichiers ne le sont pas. **Cette fonctionnalité n'est donc pas une comparaison des noms de fichiers. Il s'agit d'une comparaison du nombre de fichiers et du contenu entre deux dossiers.**

Les utilisateurs qui sont limités à la v3.2.0 peuvent voir les résultats affichés dans une grille d'affichage qui a diverses options de clic droit. Notez cependant que la v3.2.0 avait un bogue qui alignait mal les lignes si le nombre d'un dossier était plus grand que l'autre. Une non concordance a toujours été signalée correctement, mais les résultats, s'ils sont

enregistrés, seraient hors de propos. Cela a été corrigé dans la v3.3.0. Depuis la v3.3.0, il existe une myriade de nouvelles options clics droits dans la grille d'affichage, pour permettre à l'utilisateur de vérifier les discordances en fonction du nom, du hachage, hachages en double, hachages manquants, noms de fichiers manquants, etc. Voir l'illustration ci-dessous

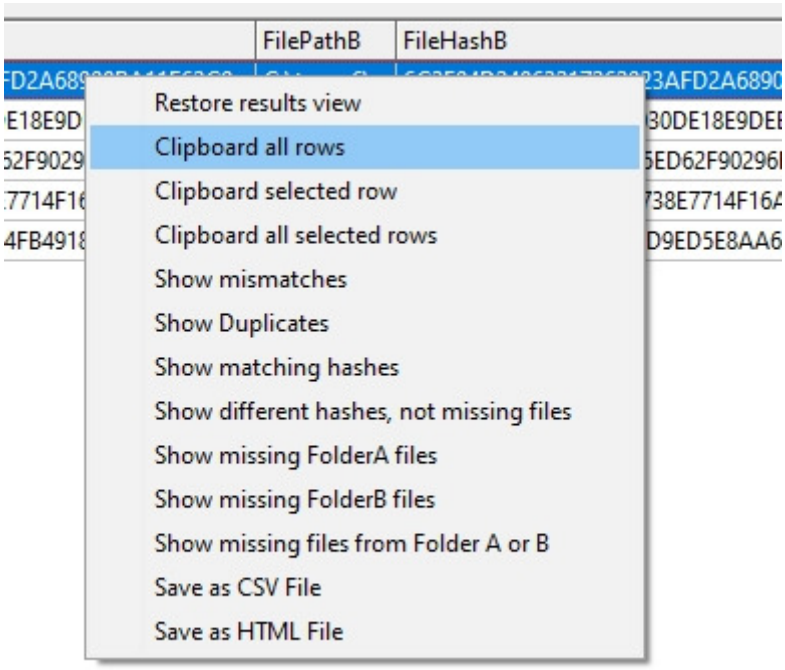


Illustration 14 : Le nouveau menu contextuel de la v3.3.0 pour Comparer deux dossiers

Si, cependant, il y a une incompatibilité de hachage même si le nombre est le même, alors continuez pour comparer le hachage de chaque fichier pour déterminer ceux qui diffèrent dans le dossier A au dossier B . Les résultats sont enregistrés par défaut (à moins que l'utilisateur ne décoche *Log results?*) dans un fichier texte et pour des résultats plus sensibles le fichier peut être enregistré par l'utilisateur en cliquant avec le bouton droit de la souris et en choisissant *Save as CSV* ou *Save as HTML File*, comme vu ci-dessus.

Notez donc qu'il compare le contenu binaire de deux dossiers en fonction des fichiers à l'intérieur de chacun. Cela ne fait pas comparer la structure de répertoire ou la structure de nom de fichier de chacun. Donc, vous pourriez avoir Dir A avec 1000 fichiers répartis dans 5 sous-dossiers différents, mais si ces mêmes 1000 fichiers sont tous à la racine de Dir

B sans sous-dossiers du tout, QuickHash signalera une correspondance, car le contenu du fichier (nombre de fichiers et les hachages de fichiers) des deux dossiers sont les mêmes.

Vous trouverez ci-dessous quelques captures d'écran illustrant la fonctionnalité v3.3.0 *Compare Two Folders*. Le dossier TestA avait 100 fichiers créés dedans, et TestB en contenait alors une copie. Notez que le nom de fichier est listé une fois, parce que sa correspondance correspondante a été trouvée dans TestB (comme décrit ci-dessus un peu plus haut dans ce paragraphe). Voici à quoi cela ressemble si les deux dossiers correspondent :

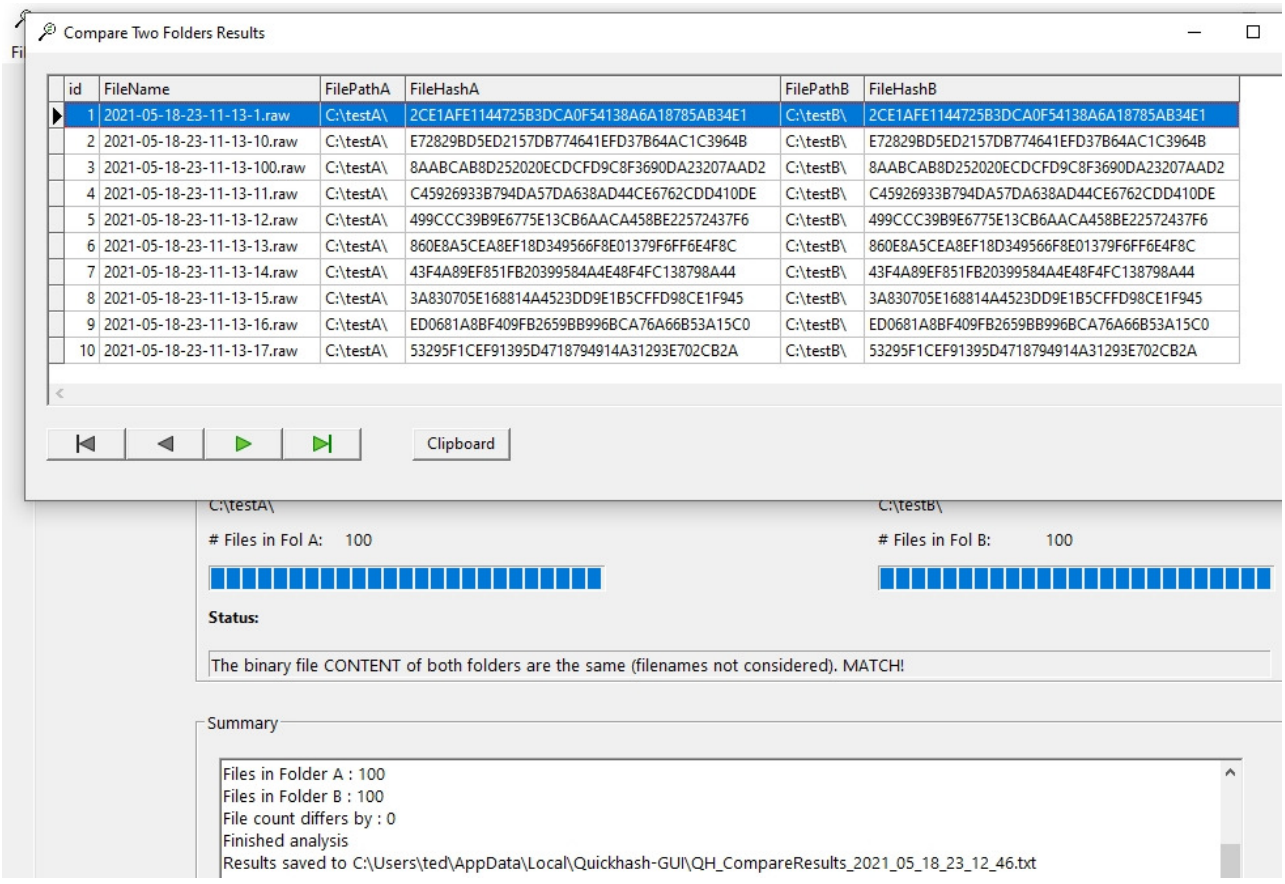
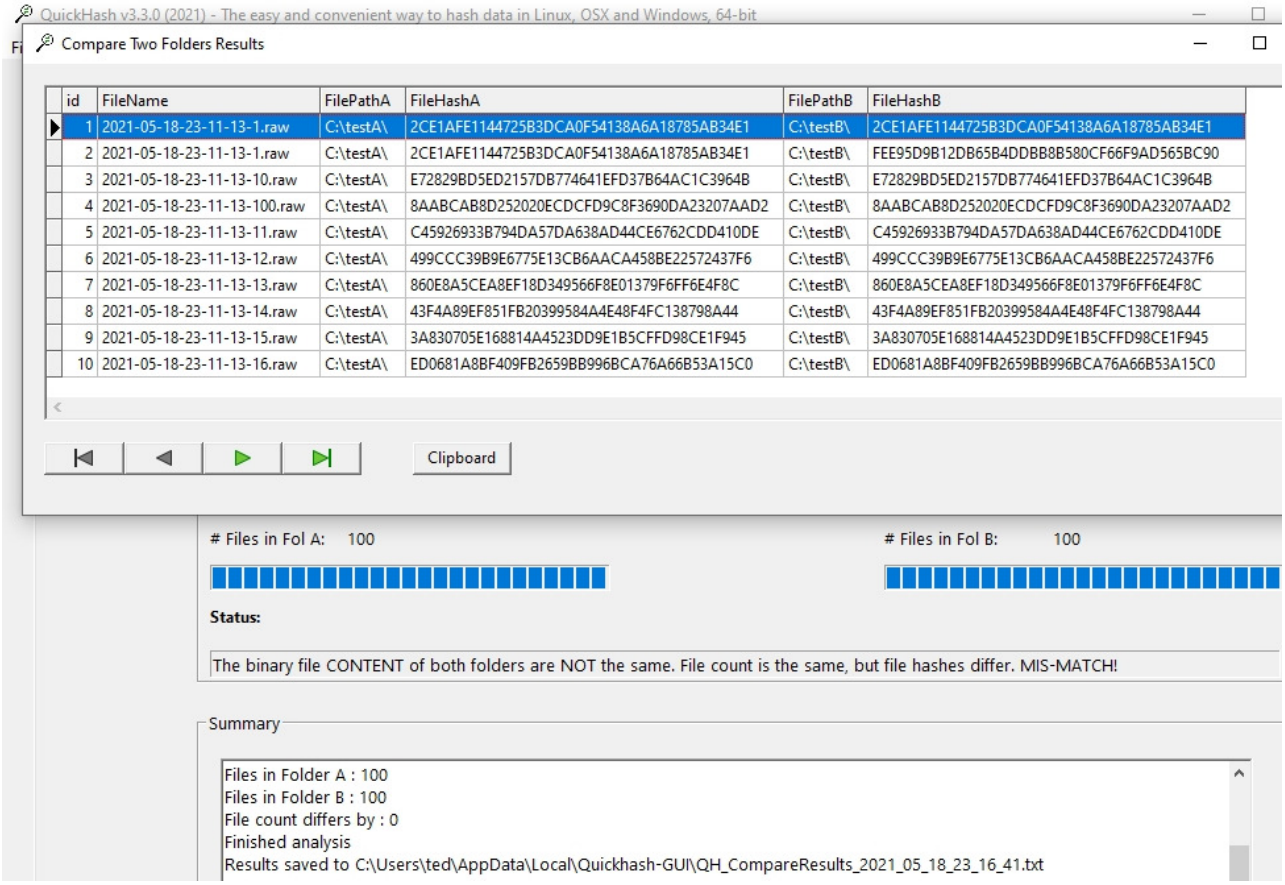
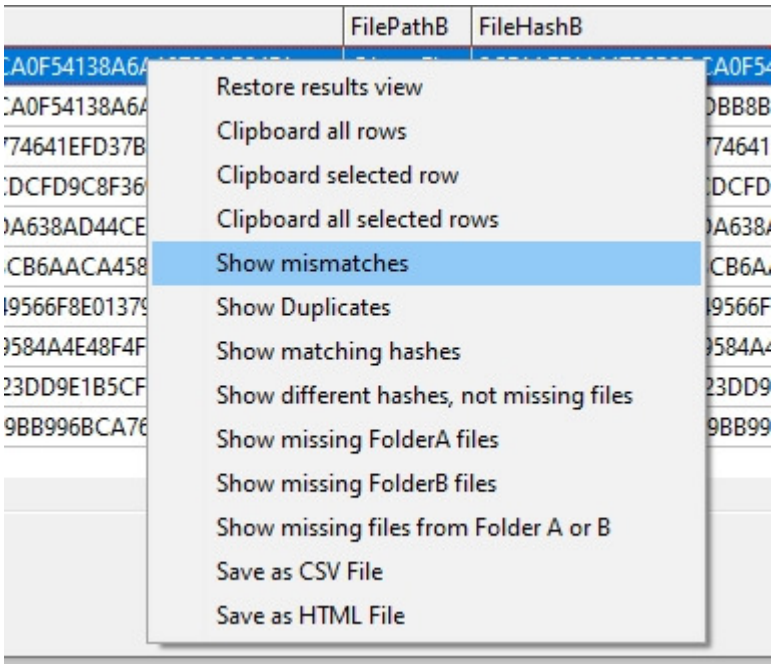


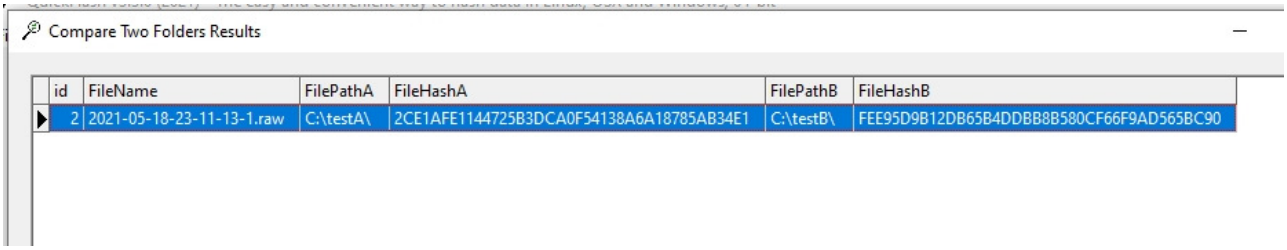
Illustration 15 : Une comparaison de correspondance standard de deux dossiers

Changeons maintenant un fichier dans TestB mais le nombre de 100 fichiers et les noms restent les mêmes :

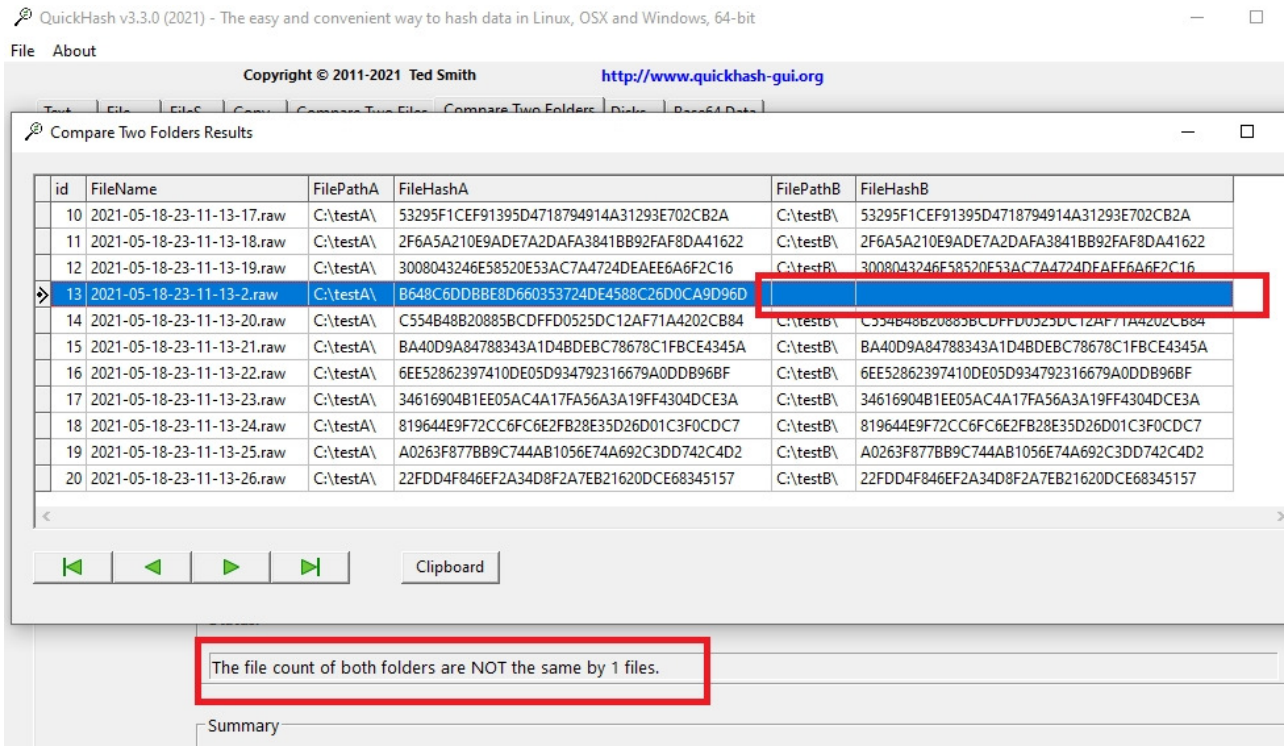


Essayons maintenant de trouver les différents fichiers en utilisant les options du clic droit :

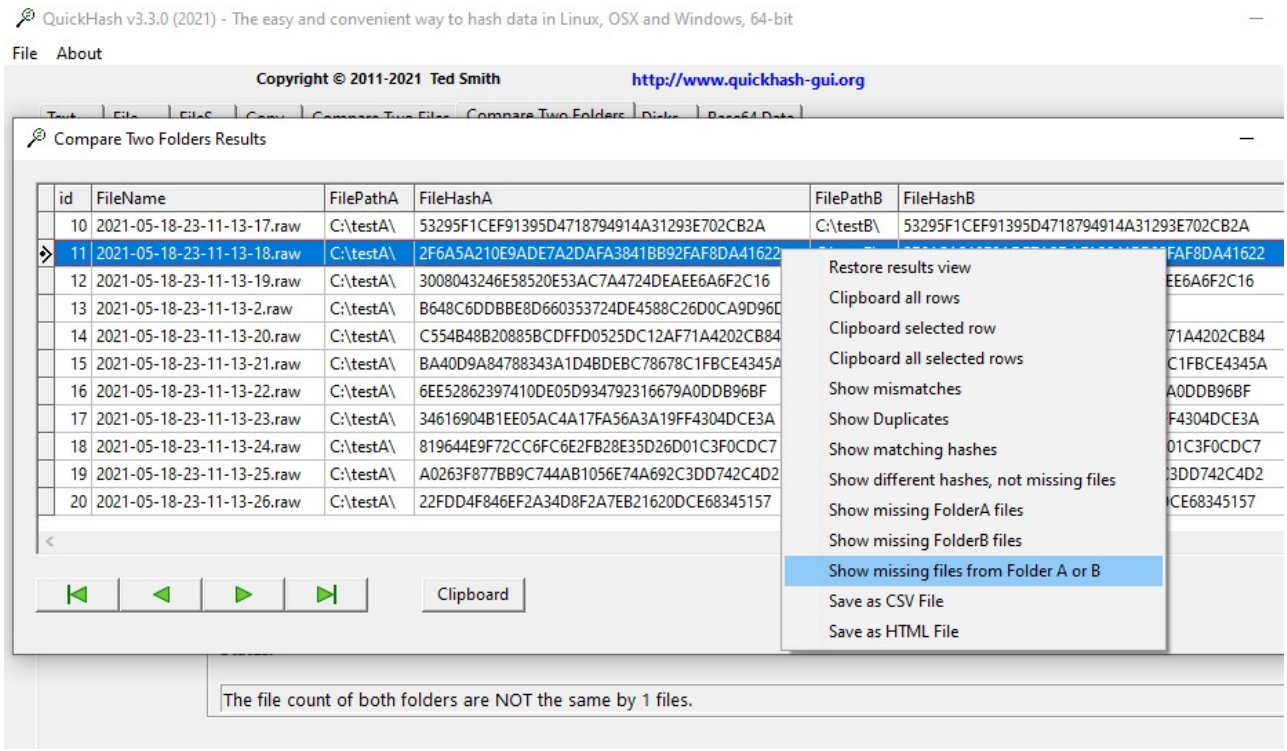


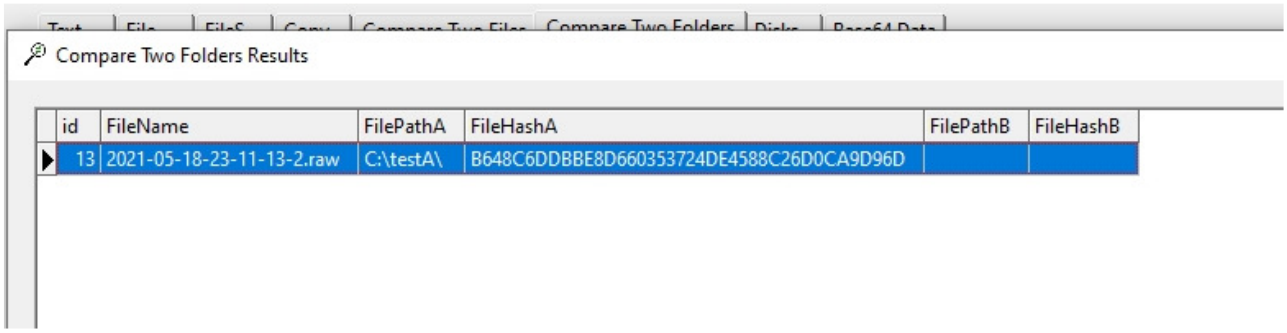


Supprimons maintenant un fichier de TestB et laissons l'autre fichier que nous avons également modifié



Voyons maintenant ce qui manque en utilisant un filtre de clic droit





... et on tourne en rond ainsi de suite.

Ce n'est peut-être pas parfait, mais c'est un assez bon système. Considérez que l'intention derrière cet onglet est d'aider les utilisateurs à confirmer, ou fournir une meilleure indication, que deux dossiers correspondent, peut-être après un exercice de migration de données ou quelque chose de similaire. Ou que deux dossiers sont similaires à 99 % avec une variance de 1 %. Il n'est pas conçu pour aider les utilisateurs à consulter deux dossiers différents à 99 %, mais simplement pour aider à identifier le 1% c'est pareil. Si tel est votre besoin, vous avez besoin d'un gestionnaire de fichiers. Pas un hacheur de données. Encore une fois, si c'est votre besoin, d'autres outils peuvent mieux vous aider.

Une fois terminé, le fichier journal est automatiquement enregistré dans un emplacement jugé sûr pour les utilisateurs de système d'exploitation particulier. Ce chemin peut varier, mais l'utilisateur est informé dans la fenêtre *Summary* en bas, où il peut alors y accéder et l'ouvrir avec n'importe quel éditeur de texte.

Veuillez noter que la fonctionnalité *Compare Two Folders* est extrêmement délicate à développer dans une façon qui convient à tout le monde. La V3.3.0 a connu la plus grande refonte de cette section et a pris de nombreuses semaines pour se développer au stade où elle se trouve dans cette version. Si, malgré les nombreuses options qui sont maintenant disponibles ça ne répond toujours pas à votre attente, alors veuillez utiliser quelque chose qui répond à vos besoins. Il existe de nombreux outils de « comparaison de dossiers » ; certains commerciaux, certains gratuits. *Directory Opus* est celui qui est fortement recommandé, ainsi que *Beyond Compare* et bien sûr le terminal Linux. Il y a beaucoup plus de choix. Je ne peux vraiment pas faire beaucoup plus que ce que j'ai, et une grande partie de cela a été réalisé avec nos remerciements à la communauté open-source et aux différents ninjas SQL.

3.1.7 Disques (Disks) :

Cet onglet n'était disponible que dans la version Windows antérieure à la v2.7.0 mais depuis lors, il est disponible pour les utilisateurs Windows et Linux. Apple Mac OSX n'est pas pris en charge actuellement.

Les utilisateurs Windows et Linux doivent exécuter QuickHash en tant qu'administrateur ou (sous Linux) root ou sudo.

La fonctionnalité permet le calcul d'un hachage pour le disque physique ou le volume logique de l’ordinateur (comme Drive E :). Utile pour comparer la valeur calculée par un outil forensique à un autre outil. Sur les disques modernes, des vitesses comprises entre 7 Go et 14 Go par minute ont

été observées. Pourtant des vitesses de 4 Go par minute sont assez courantes et toujours rapides par rapport à de nombreux autres outils. Avec l'Algorithme xxHash nouvellement ajouté, des vitesses de 15 à 20 Go par minute doivent être atteintes.

Pour utiliser la fonctionnalité, l'utilisateur doit cliquer sur le bouton *Launch Disk Hashing Module* dans l'onglet *Disks* et il sera alors affiché l'écran suivant

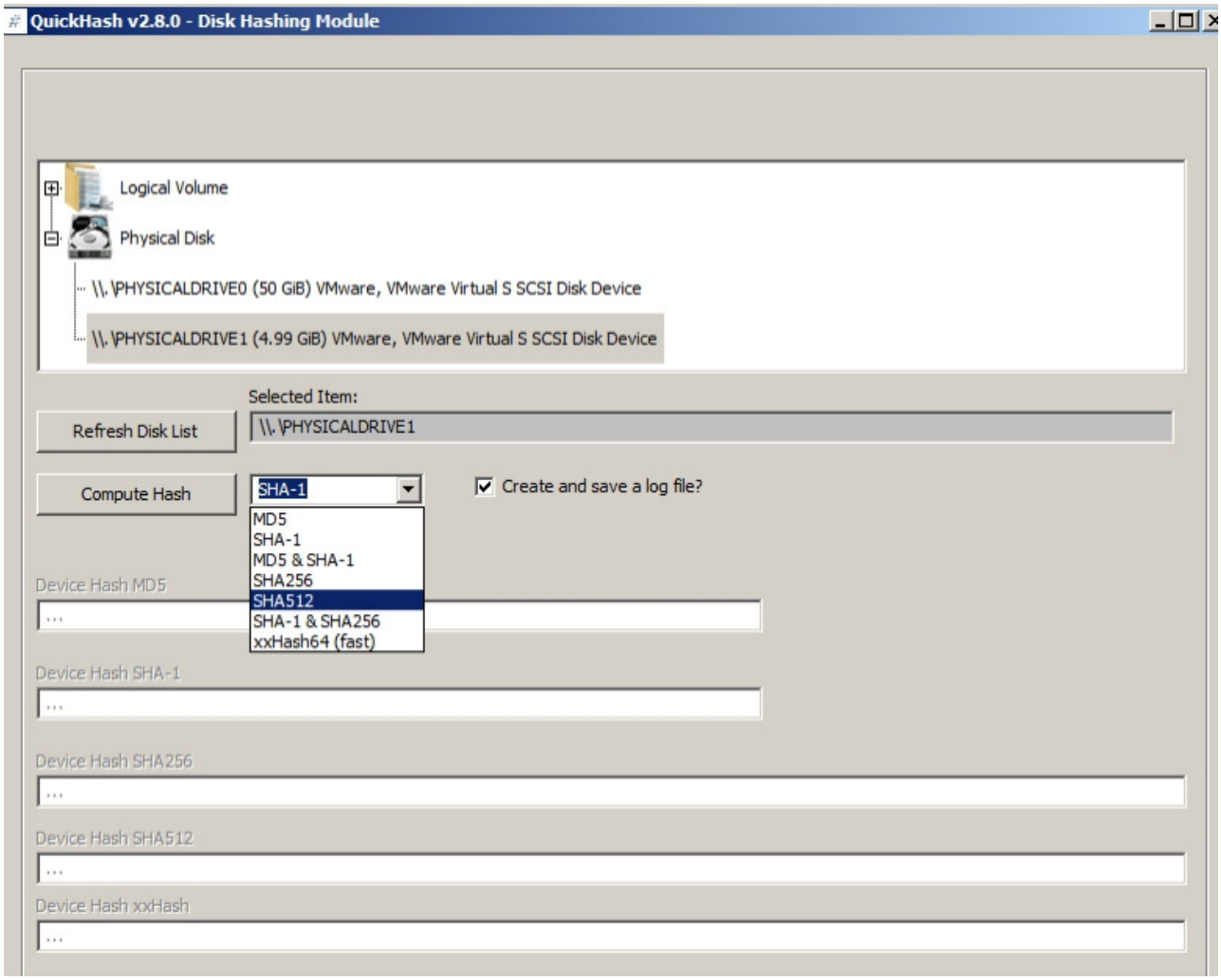


Illustration 16 : Le module de hachage de disque de QuickHash v2.8.0

Depuis la v2.8.0, une fonctionnalité de traçage complète est disponible qui enregistre la version de QuickHash, la date et l'heure de début et de fin avec le temps pris, les hachages calculés et ainsi de suite, écrits dans un fichier texte à un endroit choisi par les utilisateurs.

Le module de hachage de disque est largement basé sur le projet frère de QuickHash appelé YAFFI ('Yet Another Free

Forensic Imager') également par Ted Smith et aussi open-source, mais il n'est plus maintenu (sous réserve de modifications).

Pour les utilisateurs Windows et Linux uniquement : Pour choisir un disque, l'utilisateur doit simplement cliquer une fois sur le disque ou le volume logique, sélectionnez leur algorithme de hachage préféré (SHA-1 par défaut) puis cliquez sur *Compute Hash*. Depuis la v3.2.0, il est possible de calculer SHA-1, ou MD5, MD5 & SHA-1 ensemble, SHA256, SHA-1 et SHA256 ensemble, SHA512, xxHash (32 bits sur x86), xxHash64 (64 bits sur x64) et Blake3.

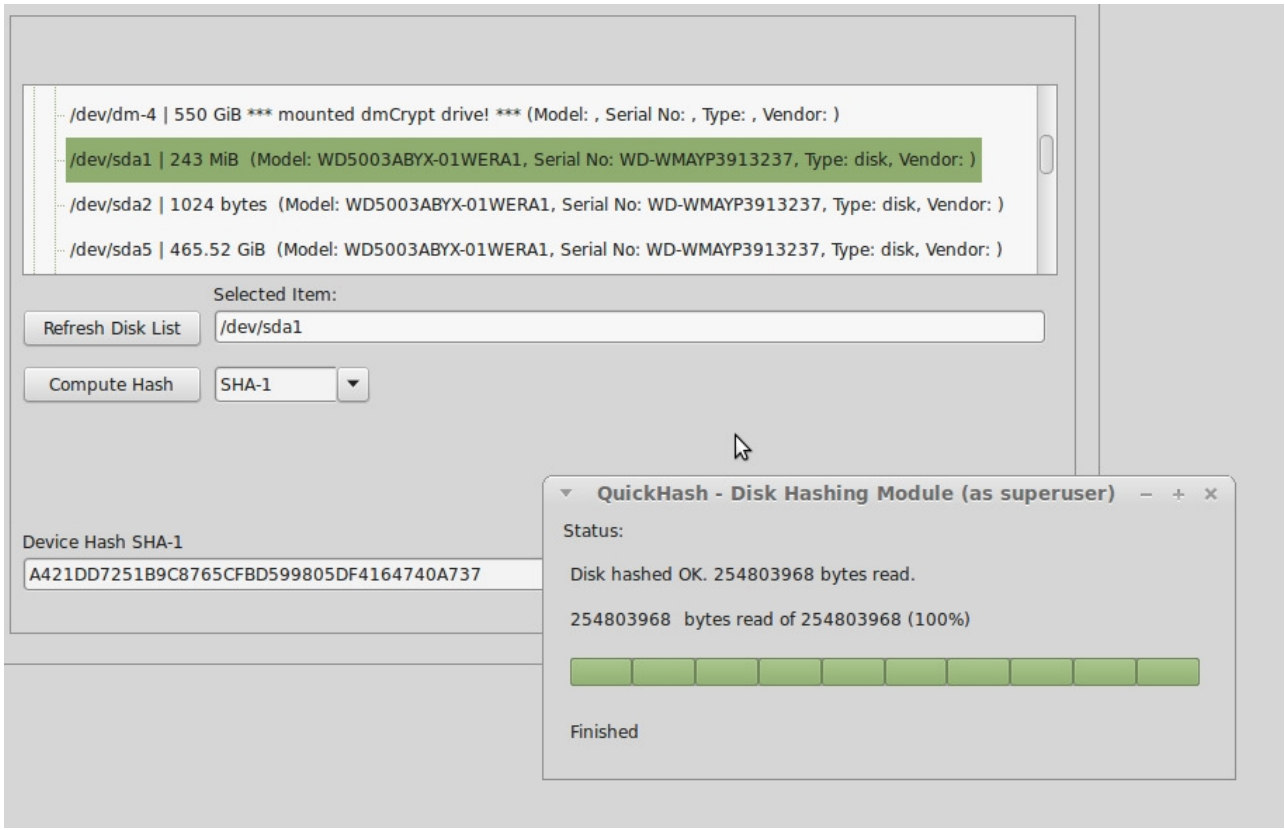


Illustration 17 : Hachage d'un volume logique sous Linux à l'aide de QuickHash

Si l'utilisateur reçoit une erreur *could not convert variant of type (Null) into type Int64*, cela est probablement dû à l'existence d'une baie périphérique de lecteur amovible installée (ceux qui permettent la lecture de la carte SD, les ports USB supplémentaires, etc.) qui sont généralement présents, ou sous MS Windows une liste de lecteurs logiques comme *Removable Drive X:* dans l'Explorateur Windows, même s'ils sont vide. L'erreur apparaît donc car ces entrées apparaissent comme un lecteur valide, mais sans aucune

capacité de disque, et cela provoque l'erreur. Des tentatives ont été faites, notamment dans la v3.0.1 et surtout dans la v3.3.0 lorsque ce code de module a été en grande partie réécrit, pour réduire cela, mais il peut encore se produire. Il est très difficile à déboguer car il est spécifique au matériel.

3.1.8 Base64 Data

Nouveau dans la v2.8.3, l'onglet *Base64* permet à l'utilisateur de hacher un fichier Base64 encodé ET de générer un hachage de son homologue décodé sans que l'utilisateur ait à créer d'abord la version décodée. Cela peut être fait pour un seul fichier, ou bien l'utilisateur peut sélectionner un dossier plein de fichiers encodés en Base64 et QuickHash générera des valeurs de hachage codées et décodées de tous. Les résultats sont affichés sur une grille d'affichage, à partir de laquelle l'utilisateur peut cliquer avec le bouton droit pour copier des lignes simples, toutes les lignes ou enregistrer la totalité de la grille.

Il y a un troisième bouton *Decode and Save files...* qui, s'il est cliqué, demandera à l'utilisateur un dossier de Fichiers encodés en Base64, puis QuickHash demandera un deuxième dossier pour y mettre les versions décodées. Il décodera ensuite tous les fichiers encodés en Base64 et enregistrera les nouvelles versions décodées dans le dossier de sortie. Aucun hachage n'est effectué ici. C'est juste un moyen rapide et facile pour les utilisateurs de décoder leurs fichiers Base64 sans avoir à utiliser des systèmes en ligne. Si les utilisateurs souhaitent que les fichiers soient également hachés, soit dans leur forme encodée ou décodée, alors utilisez la deuxième option de l'onglet pour calculer ces valeurs comme décrit (deuxième bouton *Decode and hash files*) ou bien utilisez l'onglet *File* pour en faire un à la fois (choisissez soit le fichier encodé, soit la version décodée si vous l'avez décodé), ou bien l'onglet *FileS*.

3.2 Unicode

Sachez que QuickHash est compatible Unicode sur les systèmes Linux, Apple Mac et Windows. Il traitera les fichiers avec des caractères Unicode dans leurs noms de fichiers ou dans leur contenu sans difficulté. *Noter que les versions de Windows antérieures à 2.3 n'étaient pas compatibles avec Unicode.*

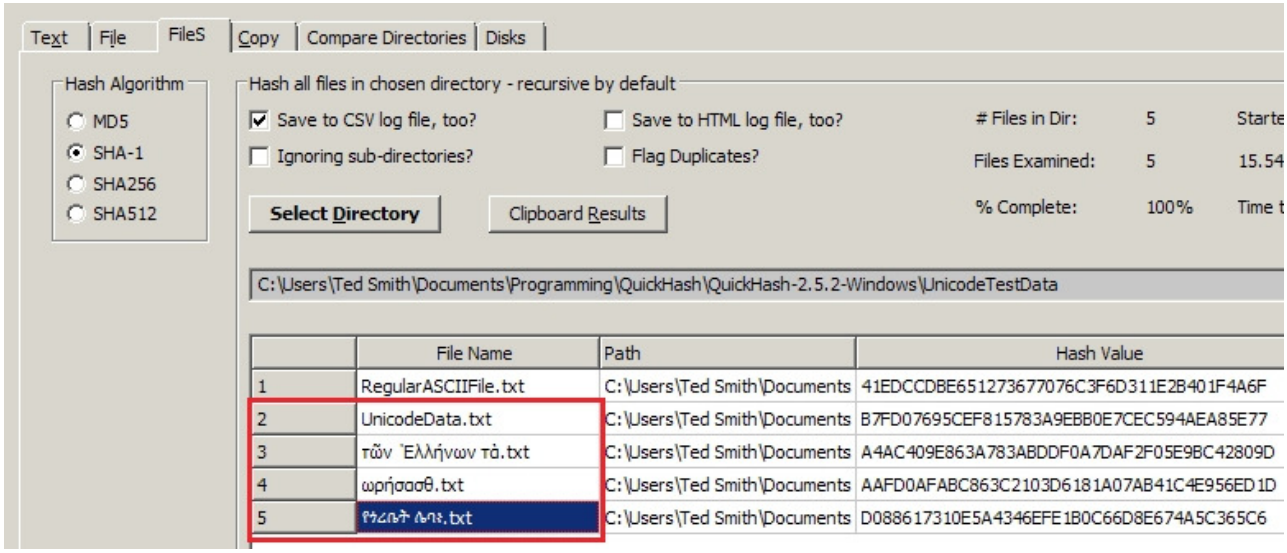


Illustration 18 : QuickHash affichant la reconnaissance d'Unicode sous Windows

3.3 Fichiers ouverts

Depuis la v1.0 de QuickHash en 2011, les fichiers qui ont été ouverts et non partagés par le système d'exploitation ont provoqué des erreurs d'accès refusé avec QuickHash.

Cependant à partir de la v3.0.1, le problème a heureusement été réduit grâce à une meilleure gestion des exceptions. À partir de la version v3.0.1, si le programme reçoit un fichier qui est ouvert par un autre programme, le programme va simplement renvoie une valeur de hachage indiquant "n'a pas pu accéder au fichier" et passe au fichier suivant, plutôt que affichant un message d'erreur et forçant l'utilisateur à abandonner le programme.

Le hachage des fichiers ouverts n'est de toute façon pas judicieux car le hachage changera probablement lorsque le fichier sera fermé, mais dans toutes les versions

précédentes, le programme ne pouvait tout simplement pas contourner le statut ouvert et plantait. Cette solution est considérée comme plus favorable.

3.4 Bibliothèques

La v3.3.0 est désormais livrée avec un sous-dossier appelé "libs". Pour Windows et Linux, cela doit être conservé dans le même endroit où QuickHash est exécuté. Sous Windows, le dossier contient la DLL suivante :

Les hachages SHA-1 pour la v3.3.0, à partir de mai 2021, sont :

libewf-x64.dll
5D33227712DA76316613DCD88B2749916DBA5ACA

libewf-x86.dll
915E3F26E170A062312A8CD73462AE6ECA6EF7BA

libgcc_s_dw2-1.dll
201924954A5A593C4CA24EE0FE799A764B41598D

libwinpthread-1.dll
34E84ED8F69F05FCAD212B02C2B064A5C7377904

sqlite3-win32.dll
0B25A2BA06DD5B8FE2A5F33C5C8442D4C12A2B70

sqlite3-win64.dll
2092405B3755C71E12E2F6EE4D193B321999CB62

zlib1.dll (32 bits)
B1D1FECBB568EDCF712232738BA3805B47BC6036

zlib1.dll (64 bits)
A10687C37DEB2CE5422140B541A64AC15534250F

Les versions Linux incluent le fichier libewf-Linux-x64.so qui est le fichier SO compilé 64 bits pour la bibliothèque libewf.

libewf-Linux-x64.so
2376C9092754ABF401CFA1D17C00801DAAB4D143

Notez que tous ces éléments sont nouveaux ou mis à jour pour la v3.3.0. Les anciennes versions de QuickHash contiendront certains fichiers de bibliothèque avec des hachages différents de ceux indiqués ci-dessus.

3.5 Le menu À propos (*About*)

Depuis quelques années, il existe un menu standard *About* pour afficher les déclarations de licence, l'utilisation des bibliothèques et reconnaître les contributions de la communauté open source. Depuis la v3.3.0, cependant, il y a deux nouvelles options de menu dans le menu À propos. La version de SQLite indiquera la version de SQLite utilisé par QuickHash. Et le vérificateur d'environnement tente d'effectuer certaines diligence autour des bibliothèques qu'il s'attend à trouver sur votre système d'exploitation choisi et il affiche également certaines données à propos de SQLite, comme le nom du fichier de base de données sous-jacent, pour le rendre plus facile à trouver, si vous en avez besoin. L'affichage des résultats diffère d'un système d'exploitation à l'autre.

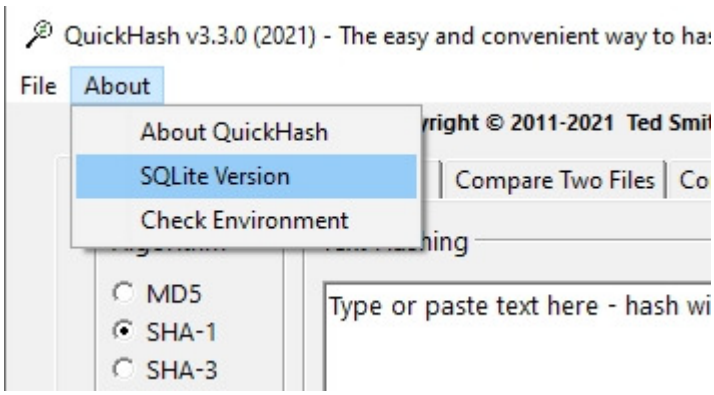


Illustration 19 : Nouvelles options de menu About avec v3.3.0

3.6 Autres outils du même nom ou d'un nom similaire

Il existe de nombreux outils de hachage disponibles - trop nombreux pour être mentionnés - et ils ont tous des atouts et des faiblesses différents, comme QuickHash. Cette section

est écrite pour essayer d'aider avec les demandes de renseignements reçus qui sont rédigés dans le sens de "J'ai téléchargé QuickHash après notre chat l'autre jour, mais il ne fait pas certaines des choses que vous avez dit qu'il pouvait ", ce qui, depuis environ 2012, a causé mes propres enquêtes, à tel point il est devenu clair que de nouveaux projets ont été développés depuis avec des noms très similaires. Ceux-ci incluent des outils exécutables en ligne et autonomes ainsi que des bibliothèques.

Il est important de souligner, cependant, que QuickHash-GUI a été le premier outil de hachage de données open source GUI, autonome, gratuit, compatible Unicode et multiplateforme nommé "QuickHash" et il a été publié sur Sourceforge en 2011 à http://sourceforge.net/projects/QuickHash

Vous ne devez pas le télécharger depuis n'importe où ailleurs.

3.7 Dons

QuickHash est créé pendant mon temps libre, qui est très étriqué avec un travail à temps plein et une famille. Il est écrit généralement tard dans la nuit, ce qui peut parfois expliquer des oublis et être à l'origine de quelques bogues ! Vous remarquerez également que le site Web est sans publicité (j'ai essayé les publicités pendant quelques mois, mais je pense ça gâche juste l'image).

Si vous appréciez le programme, ou si votre organisation, entreprise ou agence le fait, alors veuillez considérer faire un don en utilisant https://paypal.me/quickhashgui . Cela aide à financer les coûts du serveur AWS et contribue également à inspirer le développement futur. Les études de cas sont également intéressantes... si QuickHash a vous a aidé dans une tâche majeure ou à réaliser quelque chose d'important, alors veuillez envisager de soumettre un témoignage pour le site. Les deux méthodes sont une bonne occasion d'inscrire

votre entreprise sur le site Web de l'un des outils de hachage de données les plus utilisés au monde :
http://www. QuickHashgui.org

Pour faire un don :
www.paypal.me/quickhashgui